Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas Y Formales

Escuela Profesional de Ingeniería Electrónica

DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL PROTOTIPO, EN BASE A TARJETA RASPBERRY PI, PARA REEMPLAZAR LAS LABORES DE UN ENFERMERO ASISTENTE EN TIEMPOS DE PANDEMIA COVID-19, AREQUIPA 2020 - 2022.

Tesis presentada por los Bachilleres:

Benavente Salas, Luis Manuel
Chávez Aragón, Frank Edward

Para optar al Título Profesional de:

Ingeniero Electrónico con Especialidad en Telecomunicaciones

Asesor:

Ing. Sulla Torres, Raúl Ricardo

Arequipa – Perú

2022

DICTAMEN APROBATORIO DEL BORRADOR DE TESIS

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA ELECTRONICA

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 06 de Enero del 2022

Dictamen: 003295-C-EPIE-2022

Visto el borrador del expediente 003295, presentado por:

2012201691 - CHAVEZ ARAGON FRANK EDWARD
2012200661 - BENAVENTE SALAS LUIS MANUEL

Titulado:

DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL PROTOTIPO, EN BASE A TARJETA RASPBERRY PI, PARA REEMPLAZAR LAS LABORES DE UN ENFERMERO ASISTENTE EN TIEMPOS DE PANDEMIA COVID-19, AREQUIPA 2020 - 2022.

Nuestro dictamen es:

APROBADO

1547 - RODRIGUEZ GONZALES PEDRO ALEX
DICTAMINADOR

1692 - VALDIVIESO HERRERA DIANA ISABEL
DICTAMINADOR

1886 - COPA PINEDA JUAN CARLOS
DICTAMINADOR

DEDICATORIA

A mis padres Juan y Luz por haberme inculcado siempre a seguir adelante y ser una buena persona, muchos de mis logros se los debo a ustedes y este es uno de ellos, gracias, por tanto, por motivarme constantemente y hacerme conseguir mis anhelos.

A mi abuelo Alejandro por sus buenos consejos y motivarme para lograr ser una gran persona.

Benavente Salas, Luis Manuel

A Dios, primeramente, quien siempre nos guía y hace todo posible.

A todos aquellos que han sufrido la pandemia del Coronavirus. Y en memoria de todos aquellos conocidos, amigos y familiares que ya no están con nosotros, pero siempre vivirán en nuestro corazón.

A mi hermosa y maravillosa familia, que los amo con todo mi corazón.

Chávez Aragón, Frank Edward

AGRADECIMIENTO

Agradezco a Dios por iluminarme, siempre darme sabiduría y la perseverancia de no rendirme.

Merecen un gran reconocimiento especial mi madre y mi padre, por su gran esfuerzo y dedicación, que siempre me dijeron "si se puede", y me incentivaron a lograr culminar mi carrera universitaria.

A mi hermano por darme siempre el apoyo emocional, las ganas de seguir adelante.

A mis amigos en especial a mi amigo Frank, por apoyarnos ambos y para poder culminar un gran reto que nos propusimos.

Benavente Salas, Luis Manuel

A Dios, por estar a mi lado siempre y por haberme dado experiencias y personas tan maravillosas.

A mi familia, en especial a mis padres Jorge y Diana que siempre me han apoyado en todo momento, sobre todo en tiempos duros de pandemia.

A mis docentes de Ingeniería Electrónica, que desarrollaron en nosotros nuevos conocimientos y nos impulsaron con su carisma y buen humor a ser mejores cada día. Y también a mis maestros que me inculcaron el amor por el Arte en todas sus ramas.

A mi amigo Luis, por darme ánimos e incentivarme a desarrollar esta tesis para culminar nuestra carrera profesional como es debido.

A mis amigos que los quiero como hermanos, los que de una manera u otra me han aliviado el malestar emocional que he sufrido durante la pandemia. A todos y cada uno de los que fueron parte de este logro y los que dieron más luz a mi vida.

Chávez Aragón, Frank Edward

RESÚMEN

En este estudio se diseñó e implemento un prototipo de un robot móvil en base a tarjeta RASPBERRY PI 4, para reemplazar las labores de un enfermero asistente en tiempos de pandemia Covid-19, Arequipa 2020 - 2022, dentro de instituciones del sector salud. Este prototipo cumple la función de medir presión arterial, oxigenación, ritmo cardíaco y temperatura, así como llevar material médico de un ambiente a otro; pero siendo monitoreado a distancia través de sensores que le permitían recoger información tanto del paciente como del entorno y transmitirla a los operadores, evitando exposiciones al contagio. Además, realiza un cuestionario de síntomas Covid-19, el cual, es respondido mediante 2 botones para afirmación o negación. En cuanto al desplazamiento, dispone de una cámara en la parte inferior que detecta una guía de color azul, que toma como referencia para llegar a su destino. Por otro lado, tiene una cámara adicional en la parte superior para detectar al paciente, mediante reconocimiento facial. Este robot enfermero es activado remotamente por personal de salud, que recibe información del estado de salud de los pacientes, brindada por el prototipo.

Palabras Claves: Robot, RASPBERRY PI 4, pandemia, presión arterial, oxigenación, ritmo cardíaco, temperatura, distancia, cuestionario, reconocimiento facial

ABSTRACT

In this study, a prototype of a mobile robot based on the RASPBERRY PI 4 card was designed and implemented, to replace the work of an assistant nurse around Covid-19 pandemic times, Arequipa 2020 - 2022, within institutions of health sector. This prototype fulfills the function of measuring blood pressure, oxygenation, heart rate and temperature, as well as carrying medical material from one environment to another; but being remotely monitored through sensors that allowed it to collect information from both the patient and the environment and transmit it to the operators, avoiding exposure to contagion. In addition, it performs a Covid-19 symptom questionnaire, which is answered by means of 2 buttons for affirmation or denial. As for movement, it has a camera at the bottom that detects a blue guide, which it takes as a reference to reach its destination. On the other hand, it has an additional camera at the top to detect the patient, through facial recognition. This nursing robot is remotely activated by health personnel, who receive patients' health status information, provided by the prototype.

Keywords: Robot, RASPBERRY PI 4, pandemic, blood pressure, oxygenation, heart rate, temperature, distance, questionnaire, facial recognition.

INTRODUCCIÓN

Terminando el año 2019 en China, apareció una nueva enfermedad llamada COVID 19. En un principio, se pensaba que no iba a progresar mucho; sin embargo, poco a poco se fue extendiendo por todo el país y luego a varios países, hasta convertirse en una pandemia que terminaba por matar a muchas personas. Rápidamente, se pensó en varias formas de combatir este nuevo mal, y cómo evitar que sigan los contagios. Es así, que, recurriendo a la tecnología, se programaron robots como asistentes médicos que reemplazarían a asistentes humanos. Entonces, ¿Por qué no hacer lo mismo en nuestra ciudad de Arequipa?

La pandemia por COVID 19, ha avanzado bastante en el 2020 y aún sigue en este 2022. A pesar de los cuidados que debe tener cada persona y las vacunas que están surgiendo, aumentan contagios y muertes; incluso, están apareciendo nuevas variantes del virus. Pero un robot que sea manipulado a distancia es inmune a enfermedades humanas.

La realización del diseño y posterior implementación de un robot móvil, en tiempos de pandemia, logrará evitar el contacto directo entre médicos y pacientes; previniendo más contagios que propaguen la enfermedad en la ciudad de Arequipa; y de manera consecuente, proteger al personal de salud; del cual, día a día, se sufren bajas que disminuyen la capacidad de atención.

Es por este escenario, que el presente trabajo de investigación se desarrolla a causa de la ausencia de personal médico dirigido a la atención de enfermos con Covid-19,

lo que ocasiona que en muchos casos se pierdan vidas, al no recibir, en el momento oportuno, los cuidados requeridos para su tratamiento y mejoría. De igual manera, la investigación surge por la ausencia o escasez de nuevas opciones tecnológicas, que apoyen y mejoren la atención del sector salud al pueblo arequipeño. Por tanto, mediante la implementación del diseño propuesto, se evitaría el contacto de profesionales de salud con pacientes con Covid-19; reduciéndose entonces el número de contagios y defunciones que se percibe en la actualidad dentro de la plana de profesionales de salud; así como proporcionar la posibilidad de estar preparado ante nuevas crisis epidemiológicas que podrían acontecer en el futuro.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO I

GENERALIDADES DE LA INVESTIGACIÓN

1.1. Identificación del problema

Hay un alto nivel de contagio de la enfermedad producida por el nuevo coronavirus, SARS-CoV-2, debido a la exposición a la que se enfrentan constantemente los profesionales de la salud, específicamente los enfermeros que tienen como función hacer la recepción, triaje y asistencia a los pacientes que se presume o portan el virus. Además, hacen falta dispositivos inteligentes que puedan reemplazarlos en ciertas funciones, de manera que eviten el contacto directo con el paciente. Siguiendo la línea histórica de la humanidad, sabemos que la aparición de epidemias tiene períodos cíclicos, desde la peste del Justiniano hasta el COVID 19. Por tanto, sería de mucha utilidad pensar en crear dispositivos que puedan asistir a los pacientes sin exponer a otras personas.

1.2. Descripción del problema

La totalidad de los centros hospitalarios de la provincia de Arequipa, han colapsado a causa de la pandemia del Covid-19, escenario que ha ocasionado estado de emergencia sanitaria. Aun así, al igual que los pacientes, el personal de salud también se contagia y fallece; situación que requiere de un asistente no biológico que apoye a los profesionales, en cuanto a la precisión de resultados obtenidos de pacientes con Covid-19.

Por tanto, se plantea diseñar un sistema electrónico que se desplace automáticamente (siguiendo una guía de color azul) hasta encontrar al paciente (mediante la detección de rostros). Que evalúe sus síntomas con un cuestionario y pueda tomar signos vitales clave: presión arterial, oxigenación, ritmo cardíaco y temperatura, así como también llevar material médico de un ambiente a otro; todo siendo activado remotamente. Dicho sistema, a través de sensores recogerá información tanto del paciente como del entorno y la transmitirá a los operadores, evitando exposiciones al contagio.

1.3. Alcances de la investigación

En el presente estudio, se realizará el diseño físico del prototipo de robot móvil en base a tarjeta RASPBERRY PI 4; al igual que la implementación de dicho dispositivo en los centros de salud de la provincia de Arequipa.

1.4. Objetivos de la investigación

1.4.1. Objetivo general

Diseñar e implementar el prototipo de un robot móvil en base a tarjeta RASPBERRY PI 4, que reemplace las labores de un enfermero asistente en tiempos de pandemia Covid-19, Arequipa 2020 - 2022, dentro de instituciones del sector salud.

1.4.2. Objetivos específicos

1) Diseñar el mecanismo de desplazamiento del robot controlado remotamente.
2) Diseñar mecanismo de monitoreo para medir presión arterial, oxigenación, ritmo cardíaco y temperatura a los pacientes.
3) Diseñar un sistema de comunicación a distancia (incluyendo una cámara) entre el paciente y personal de salud en la PC principal.
4) Establecer el método de comunicación WIFI entre la PC principal y el robot móvil.
5) Diseñar una aplicación que muestre al médico un registro de las mediciones tomadas a cada paciente.
6) Implementar un espacio en la estructura del robot, para que éste lleve material médico con 1kg de peso.
7) Construir la estructura base prototipo (considerando dimensiones menores a las de un humanoide).

8) Construir el mecanismo de desplazamiento y monitoreo del robot móvil.
9) Ensamblar los dispositivos necesarios para la comunicación vía Wifi con el robot móvil (envío de datos).
10) Realizar programación para el desplazamiento y monitoreo del robot móvil.
11) Realizar programación para el enlace cámara <-> PC vía Wifi.
12) Realizar la programación de la aplicación que permita enviar datos adquiridos de los pacientes a la PC principal.

CAPÍTULO II

MARCO TEÓRICO

2.1. Estado del arte

Para América Latina, la innovación en cuestión de robótica, apenas empieza, en comparación con el uso de la robótica en Europa, Asia y EE.UU. (Cornejo, Cornejo-Aguilar, & Perales-Villarroel, 2019) hacen una revisión a la evolución del tema en América Latina y Perú, la robótica como herramienta para el diagnóstico, pronóstico y tratamiento del paciente; uno de los retos más grandes con el uso de la robótica aplicada a la ingeniería médica y sanitaria, es el nivel de riesgo en la atención al paciente, ya que por sobre todo se debe garantizar la confianza y comodidad del paciente, razón quizás por la que muchos centros médicos descartan por ahora esta opción. Sin embargo, las necesidades se hacen latentes de forma cotidiana en los campos de cirugía y rehabilitación, y de forma coyuntural, en la asistencia de pacientes frente a la COVID-19. Analicemos por partes, primero, en el ámbito de aplicaciones quirúrgicas y de rehabilitación, se considera la siguiente secuencia y articulación de conceptos y tecnologías.

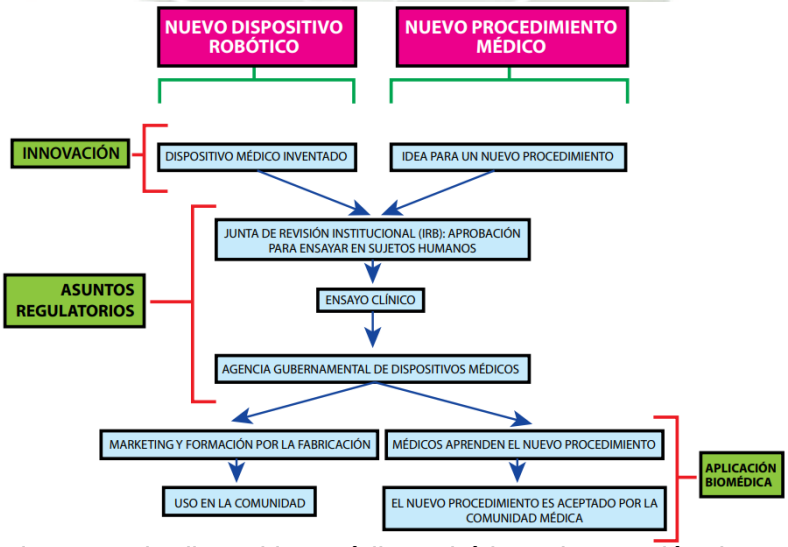


Figura 1. *Diagrama de dispositivo médico robótico e innovación de procedimientos.*

Fuente: Cornejo (2019).

Más allá del factor confianza y comodidad respecto al uso de robots en el campo médico, lo que se busca es mejorar los resultados; actualmente los equipos y dispositivos utilizados son importados, algunos de los países que vienen innovando más son México y Brasil; en el ámbito de cirugía, las especialidades donde se utiliza más son cirugía general, urología y ginecología, debido a su eficiencia al momento de la cirugía en comparación con el procedimiento manual, con mayor precisión y destreza, y sin problemas de fatiga, reduciendo el tiempo de recuperación, pérdida de sangre, tiempos menos extensos de intervención, y pudiendo ser autónomos, tele operados, montados en mesa o montados en el paciente, en serie, paralelo o móviles, (el Da Vinci como el sistema de cirugía más utilizado); el grado de intrusión en el cuerpo del paciente es alto, sin embargo el desarrollo de la tecnología está a la altura y garantiza excelentes resultados.

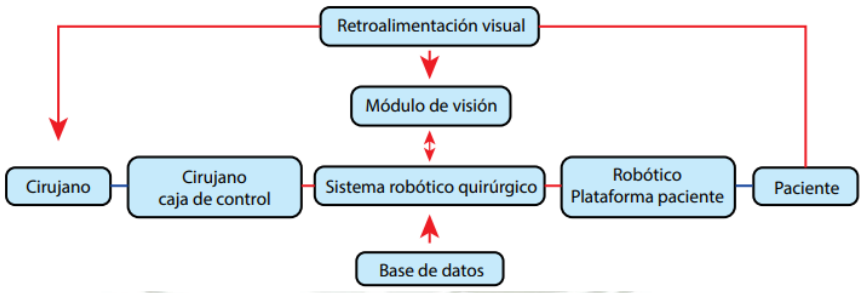


Figura 2. *Diagrama de bloques del sistema robótico quirúrgico-Control.*
Fuente: Cornejo (2019).

En Perú, los autores, han estado realizando el proyecto Biomedik Surgeon como prototipo de un sistema robótico de entrenamiento y simulación.

Y respecto a rehabilitación y asistencia, la robótica de rehabilitación es uno de los campos que implican mayor interdisciplinariedad, fusionando la mecatrónica, la neurociencia, la ingeniería eléctrica, ciencia cognitiva, procesamiento de señales, nanotecnología, ciencias de la conducta y cognitiva; utilizando la comunicación por retroalimentación y los sensores de formas novedosas. Tanto en el campo de prótesis

(sustituto de una parte del cuerpo) como órtesis (apoyo para una parte del cuerpo), en estos sistemas, la relación con el cuerpo humano no es de intrusión pero si de interacción por contacto, una interfaz hombre-máquina, llegando incluso al uso de Realidad Virtual para conceder al paciente experiencias no físicas, que generan reacciones e impulsos físicos, para ello se ha estandarizado la sistematización de secuencias y relaciones (Figura 3) y tomando en cuenta que el Perú tiene el porcentaje más alto con discapacidad (31.28%), de América Latina; se hace imperativa su aplicación e innovación. La robótica asistencial es otro campo donde la interacción del dispositivo con el paciente es directa y se utiliza con fines domésticos, de movilidad, de hospitalidad y terapéuticos.

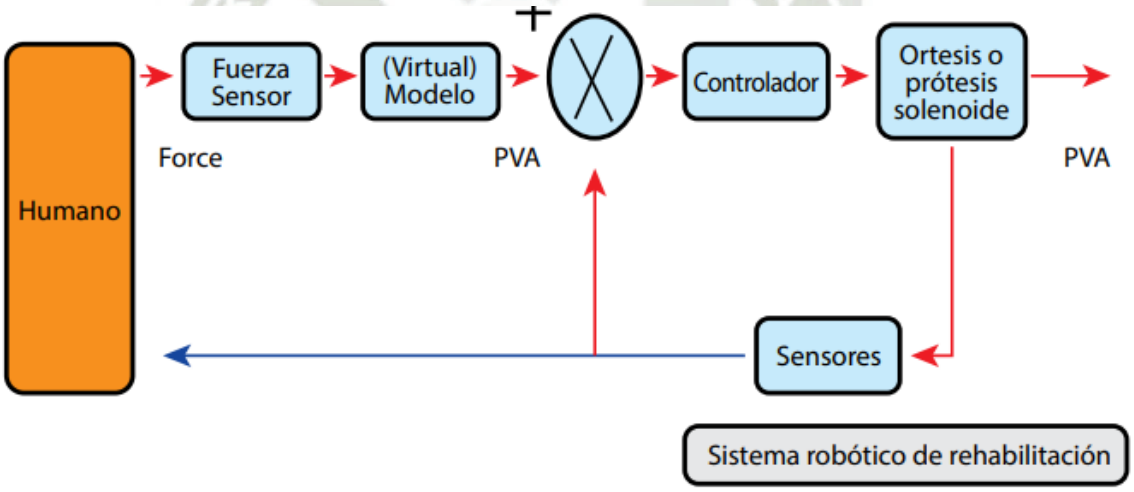


Figura 3.*Diagrama de bloques del sistema robótico de rehabilitación-Control.*
Fuente: Cornejo (2019).

En Perú existe una empresa, Biom3D, que se dedica al diseño de prótesis para niños en condición económica baja, el proyecto lleva el nombre de “Manos en Acción”.

Este es el panorama en América Latina y Perú, aquí apenas hace 15 años, que se empezó a importar tecnologías del extranjero, lo que poco a poco va motivando a los ingenieros y profesionales de rubros relacionados a experimentar y elaborar prototipos. (Cornejo, Cornejo-Aguilar, & Perales-Villarroel, 2019)

Siendo el 2022, y encontrándonos aun atravesando la pandemia por la COVID-19, surge otra aplicación de la robótica, dado que esta pandemia se origina por el contagio de un virus, de alto grado de transmisibilidad, por lo que se adoptó el aislamiento, y distancia social. Sin embargo, siempre son requeridas ciertas atenciones personales presenciales, que tienen alto grado de riesgo por la exposición al contagio, inclusive siguiendo todos los protocolos de equipo y protección; a lo que la Ingeniería Robótica y Biomédica tuvieron respuestas alternativas, destinando el uso de robots para la asistencia directa del paciente infectado, evitando que el personal médico se exponga, facilitando el monitoreo de signos vitales a través de brazos robóticos guiados de forma remota, e inclusive con dispositivos bio-mecatrónicos como la “Cápsula Endoscópica” que permite el diagnóstico no invasivo, sin contacto; así como la desinfección y esterilización de espacios a través de luz UV. Pero la robótica doméstica no se queda atrás, ya que se empleó robots móviles, para el recojo de residuos, desinfección de calles y espacios públicos, o el uso de Drones (robots aéreos), para facilitar la entrega de medicinas o alimentos. La Telemedicina dio un gran paso también, facilitando la comunicación entre comunidades a través de sistemas robóticos con capacidad autónoma de desplazamiento, facilitando el triaje, consulta médica, o capacitación preventiva a distancia; cabe recalcar que la mayor parte de estas implementaciones y aplicaciones de la robótica respecto a la pandemia se dieron en países europeos o en USA, ya que la realidad económica y administrativa en Latinoamérica, y el Perú específicamente, no consideran esa envergadura. (Cornejo, Vargas, & Cornejo-Aguilar, 2020)

2.2. Antecedentes de la investigación

2.2.1. Antecedentes Internacionales.

- Vásconez, en su trabajo de investigación titulado, “Construcción e implementación de un robot móvil usando un SBC (Single-Board Computer) para aplicaciones de transporte de insumos médicos en el centro de salud tipo

C Lizarzaburu en la ciudad de Riobamba" el objetivo de este estudio es optimizar el tiempo y trabajo del personal de un Centro de Salud, para ello previamente se hizo el estudio de los componentes electrónicos y mecánicos que conformarían el prototipo, el cual fue construido en fibra de vidrio, y mediante visión artificial logró el reconocimiento de obstáculos tanto dinámicos como estáticos, para lo que se usó un sensor Kinect ensamblado a un procesador SBC Raspberry Pi 3, logrando un desplazamiento óptimo en el prototipo en base a un sistema de control de locomoción, se sumó a este prototipo un Arduino Mega para un mejor procesamiento de datos, y la comunicación serial se logró con la tarjeta Raspberry Pi 3. La autora además propone una imagen específica para este prototipo, ya que será usado en centros de salud, la propuesta es un rostro similar al del humano, para generar familiaridad y confianza en los pacientes. Este modelo fue puesto a prueba en base a muestras no poblacionales, comprobando su capacidad de locomoción y sensibilidad de distancia, siendo el sensor ultrasónico altamente confiable para la traslación del prototipo, por otro lado el sensor Kinect presento solo 83.33% de precisión, ya que uno de sus principales inconvenientes es el ángulo de visión, respecto a las pruebas de seguidor de línea y ubicación de coordenadas se presentó solo una desviación de 2.38 cm, pudiendo además desplazarse con eficacia en cuestión a tiempo, se logró una duración de batería de 2.28hrs, es decir durante ese lapso el prototipo tuvo autonomía teórica. Por último, se procedió a una encuesta a personal trabajador del centro de salud y pacientes, sobre el grado de aceptación del prototipo de robot, se preguntó si creían que ayudaría a optimizar la atención, el 86% sostuvo que era una buena idea, el 14 % no estuvo conforme; y respecto a la apariencia del robot, al 96% les pareció amigable y al 4% no le pareció amigable. (Vásconez, 2018)

- Camacho y Escobedo, en su trabajo de investigación denominado, “Sistema robótico móvil para el transporte de materiales ligeros”, el autor propone una alternativa para transportar materiales, con un enfoque, primero doméstico. Se propone la construcción de un mecanismo simple, 1 chasis, 4 motores y elementos electrónicos. Para ello en la primera parte estudian la relación histórica entre el hombre y la máquina-robot y luego proceden a desarrollar la propuesta. Cuando el prototipo fue probado se encontró que, respecto al sensor ultrasónico, éste tiene unos 15 cm de rango de desviación. Ahora, ya que el propósito principal de este robot es el transporte de material, se calculó el peso que puede cargar, con resultados óptimos y respuestas correctas carga pesos menores de 2kg, esto debido a que el robot tiene una base metálica de unos 5 kilos. El autor concluye que se logró construir un robot semiautónomo funcional, teniendo una buena comunicación en tiempo real con el prototipo, y con un sistema bastante completo para ser aplicado en asistencia de labores, el único punto en el que no se obtuvieron resultados muy favorables fue respecto a las llantas que tenía y el peso que ralentizaba su movilidad. (Camacho & Escobedo, 2019)

- Pérez en su investigación titulada “Robot móvil con percepción auditiva y sistema de control a distancia”, realizada en el año 2019 para optar el título de Ingeniera en Computación; tenía como objetivo, implementar reconocimiento de voz a un autómata móvil, proporcionando el método de comunicación local y de comunicación remota. En cuanto, al mencionado, primeramente, a través de la conexión de un módulo de reconocimiento de voz instalado en el prototipo; y el segundo, mediante el empleo de una aplicación diseñada para establecer la comunicación entre el entorno Android con Arduino, por medio de un módulo Bluetooth. En el estudio, se emplearon sensores ultrasónicos actuantes como detectores de proximidad, a manera de otorgar al móvil la capacidad de evadir

obstáculos al instante de la navegación de su recorrido, previniendo de esta manera, deterioros en su propia estructura. Se pudo concluir, que el objetivo del estudio fue alcanzado, ya que se permitió la manipulación dual del robot móvil, inclusive fue posible alternar entre el empleo de uno u otro modo de manipulación en el transcurso del camino del robot, durante una trayectoria regular delimitada. Asimismo, para la utilización del método local, el entrenamiento de los comandos se trata de la captura y almacenamiento del sonido con intervención de un módulo de reconocimiento de voz, el cual requiere ser comparado dato a dato con cada una de las órdenes que han sido grabadas con anterioridad; volviendo dificultoso el reconocimiento de las palabras; mientras que, en la comunicación remota, escenario donde se emplea el reconocimiento de voz de Google, el entrenamiento se trata de la captura, acopio y procesamiento de la información mediante técnicas de inteligencia artificial, otorgándole un rendimiento más óptimo. (Pérez, 2019)

- Pi Bot-B: robot móvil con una Raspberry Pi es un proyecto desarrollado por Thomas Schoch y que se muestra en la Figura 4, consiste en adaptar la Raspberry Pi con un [kit de chasis Zumo de Pololu]. El Pi Bot-B se controla gracias a una aplicación de iPhone personalizada que se comunica a través de Wifi con Raspberry Pi y que se ejecuta a través del servidor web y PHP. Un programa Python envía señales a un [controlador de motor L293D] que impulsa los dos motores reductores en el chasis Zumo. La aplicación para el iPhone muestra el vídeo de la cámara web Logitech C300 y las teclas encargadas de los movimientos de Pi Bot-B. También dispone de una placa matriz de LED integrada de 8 × 8 de (Adafruit) que indica su estado. (Grayson, 2013)

Figura 4. *PiBot-B Robot móvil .*
Fuente: Grayson (2013).

El proyecto Wifi Car "JJ" mostrado en la figura, fue patrocinado por Van Leeuwen Opensource Consultoría en la campaña de Kickstarter creada por Dexter Industries. Gracias a Van Leeuwen, se respaldó el proyecto Brick Pi que fue desarrollado por Sitfit. Este proyecto que cuenta con 82 piezas se compone principalmente por una placa Brick Pi unida a una Raspberry que se controla vía Wifi. Gracias a una aplicación creada con el programa Lego digital designer. (Dexter Industries, 2014)

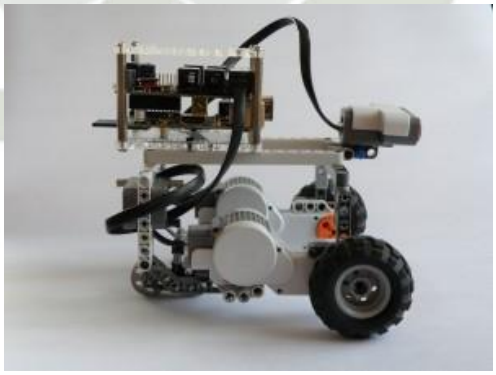


Figura 5. *Wi-Fi Car "JJ".*
Fuente: Dexter Industries (2014).

2.2.2. Antecedentes Nacionales.

- Agurto, en su tesis de investigación de Ingeniería electrónica y telecomunicaciones, denominada, “Integración de un sistema de adquisición de datos mediante el uso de un Arduino Mega y Raspberry Pi3 como servidor Web y base de datos”, tiene como objetivo crear un sistema para recolectar información del medio, y que pueda generar un reporte, que pueda ser consultado en tiempo real, a través de cualquier dispositivo en conexión a Internet. Este trabajo, pese a tener otros fines, nos interesa dado el sistema de integración usado con la tarjeta Raspberry y la comunicación entre el procesador y otros dispositivos. El autor sostiene su propuesta en la necesidad de monitorear datos e información de forma remota y sumado a ello generar el almacenamiento organizado de datos, que puedan ser consultados y procesados posteriormente, para ello integran un sistema en base a Arduino Mega, Módulo Wifi ESP8266 y el Raspberry Pi3. En el desarrollo de este sistema, Arduino sirve para la adquisición de datos, el sensor DHT11 para medir la humedad y la temperatura, los datos se envían al módulo ESP8266, Y EL Raspberry Pi3 como la base de datos y servidor web. Para ello se utilizó el software APACHE, MY SQL para la base de datos y PHPMYADMIN para administrar los datos generados, la instalación de todo el software fue hecha en esa secuencia previa actualización del Raspberry bajo los comandos, “sudo apt-get update” y “sudo apt-get upgrade”. En este sistema con componentes inalámbricos, toda la transmisión de datos es por medio del Wifi. El desarrollo, diseño de la arquitectura informática y configuración de módulo de comunicación fue óptimo, logrando la toma, recopilación y almacenamiento de datos del entorno. (Agurto, 2020)

- Flores y Garay en su trabajo de investigación denominado “Diseño e implementación de un robot móvil de desplazamiento autónomo basado en un controlador proporcional derivativo”, realizado en el año 2017 para optar el título Profesional de Ingeniero Electrónico; tenía como finalidad, realizar el diseño y la ejecución de un robot móvil a modo de controlar el desplazamiento autónomo en el plano cartesiano, a través de las coordenadas que han sido establecidas por el usuario, y cuantificada de manera algorítmica por el Software LabVIEW. Se trató de un estudio, donde se empleó el método cuantitativo tecnológico; donde el autómata posee dos motores DC en una distribución diferencial para el deslizamiento; siendo controlados por medio de un controlador proporcional derivativo, los cuales cuantifican los pulsos entregados por los encoders incrementales. En referencia a las señales independientes producidas por cada motor mencionado, son resueltas por sus microcontroladores Arduino (Slave), los cuales llevan a cabo el control proporcional derivativo. Ahora bien, los dos Arduino (Slave), tienen conexión a través del protocolo I2C al Arduino (Master) que permite realizar la programación del LIFA. Y a su vez, comunicarse con la PC de forma móvil, a través del Bluetooth. Se pudo concluir que, en el LabVIEW, los bloques de Arduino absorben gran cantidad de tiempo de procesamiento, por lo que no es posible alcanzar la velocidad más alta de adquisición, aun cuando se puede lograr información de adquisición válida; además, dichos bloques, no facilitan el diseño del algoritmo, gracias a que los mismos, comprenden funciones que evitan ejecutar diagramas más amplios y seguros. Finalmente, se recomienda emplear versiones similares de software entre el Arduino y el LIFA, a modo de prevenir incompatibilidades; al mismo tiempo, emplearse cables cortos para las conexiones de Bluetooth, y así no afectar la comunicación. Por otro lado, no resulta conveniente utilizar delay en los algoritmos de control, ya que ocasiona tiempo perdido en el microcontrolador. Diseñándose una línea de control de pequeña dimensión con

la función micros, se facilita que el microcontrolador se encuentre procesando en todo momento, sin llegar a dormir en ningún instante. (Flores & Garay, 2017)

2.2.3. Antecedentes Locales.

- Salcedo en su investigación titulada “Desarrollo de un discriminador de calidad de un sistema de control y monitoreo embebido basado en tecnología FPGA”, realizada en el año 2018 para optar el Título Profesional de Ingeniero Electrónico; pretendía llevar a cabo, el desarrollo de un discriminador de calidad de un sistema de inspección y monitoreo embebido sustentado en tecnología FPGA. En el estudio, el prototipo está comprendido de una estación, que parte de una base mecanizada para ello, y donde se ubican distintos módulos con diferentes dispositivos y elementos, que conforman una red automatizada y electromecánica. En referencia al control de la instalación, se administra y ejecuta a través del software LabVIEW, cuya lista de instrucciones y ejecución de órdenes programadas, son ejecutadas por un dispositivo embebido de costo reducido (NI myRIO), recomendada para control e instrumentación; que dispone de un procesador programable que ejecuta un sistema operativo en tiempo real, al igual que un FPGA personalizado. Se pudo concluir, que el prototipo da cumplimiento a la totalidad de la demanda requerida para su correcta operación; además de su flexibilidad al momento de su ejecución, y en el sistema de visión artificial y en el control automático. Igualmente, el prototipo da la posibilidad de eliminar de manera total o parcial, la participación del ser humano; disminuye los egresos por mano de obra directa y eleva la calidad de producción. Por último, se recomienda realizar la inclusión en el programa y calibración, de los equipos y sensores de campo instalados; a manera de asegurar la fiabilidad de sus lecturas; al mismo tiempo, de colocar mayor cantidad de sensores en los puntos donde se requieran. (Salcedo, 2018)

2.3. Bases teóricas

2.3.1. Evolución de la Robótica móvil.

El término robot, tal como expone Barrientos et al. (2007), viene manejándose desde civilizaciones pasadas, como el caso de la griega, pero comenzó su auge con la civilización árabe. La robótica surge como ciencia desde el siglo XX, hacia la década de los cuarenta; pero igual sus orígenes vienen del siglo VIII, con la edificación de autómatas humanoides por Vaucanson y los Jaquet-Droz. Ahora bien, aun cuando con el transcurrir del tiempo, se fue desarrollando gran cantidad de figuras dotadas de partes móviles, la revolución industrial en los inicios del siglo XIX fue la impulsadora del desarrollo de las distintas áreas científicas y la creación de nuevos dispositivos adaptables a diferentes rubros; los cuales luego confluyeron en el concepto de la robótica.

Así pues, en el año 1921 el escritor checo Capek, en su obra dramática Rossum´s Universal Robots (R.U.R.), batió la palabra robot, partiendo del término checo robota, que simboliza servidumbre o trabajo forzado; mientras que Asimov, implantó por primera vez la palabra robótica. Debido a los trabajos que venían desarrollándose y al auge de las obras, el término robótica acaparó la atención de los principiantes e investigadores, quienes fueron incorporándose en el estudio del concepto.

Así pues, nace el requerimiento de establecer la definición de robot, pudiendo considerarse como un sistema electromecánico reprogramable, otorga la posibilidad de ejecutar distintas funciones repetitivas que ameritan determinado nivel de precisión. Debido a que se manejan una extensa gama de formas y aplicaciones para los robots, según la función que ejecuten, resulta posible considerar distintos criterios para su categorización, como son: su arquitectura, su generación, el nivel de inteligencia, el nivel de control o el nivel de su lenguaje de programación. (Barrientos, García, & Silva, 2007)

Partiendo de la década de los años setenta, tuvo auge el crecimiento de la investigación y diseño de robots móviles; se puede mencionar a Nilsson en el SRI, que desarrolló el robot Shakey, el cual se convirtió en el primer mecanismo en utilizar la

inteligencia artificial para tener control de sus movimientos. Igualmente, destacan el Newt, desarrollado por Hollis y Hilare, desarrollado en el LAAS en Francia. En el Jet Propulsión Laboratory (JPL) se desarrolló el Lunar Rover, diseñado especialmente con objeto de la exploración planetaria.

Para fines del mencionado lapso de tiempo, Moravec desarrolló en la Universidad de Stanford, el Stanford Cart, el cual tenía la capacidad de continuar un recorrido delimitado por una línea. En el año 1983, Raibert fue desarrollado en el MIT, siendo un autómata que presentaba una sola pata, y el cual estuvo manufacturado con el objeto de estudiar la estabilidad de dichos sistemas, y a la vez, aportar mayor conocimiento acerca de dicha locomoción.

Por otro lado, se desarrolló a inicios de los años noventa, un autómata "uniciclo" (una sola rueda, similar a la de una bicicleta) en el MIT; mientras que posteriormente, en el año 1994, el Instituto de Robótica CMU desarrolló a Dante II, un sistema de seis patas, cuya finalidad de diseño era captar muestras de gases en el volcán Spurr, ubicado en Alaska. Igualmente, en el año 1996 en el CMU, se desplegó el Gyrover, un autómata que no contaba con ruedas y patas, sustentado en el funcionamiento del giroscopio, haciendo que la estrictez de sus movimientos y su fijeza sean muy altas.

Igualmente, en ese lapso, se llevó a cabo en el MIT, el Spring Flamingo, autómata que simulaba el movimiento de un Flamingo, construido para efectuar técnicas efectivas de control en la posición de los actuadores (patas), a modo de hacer una descripción del movimiento del autómata, y realizar diversos algoritmos de desplazamiento. Por su parte, en el año 1997, la NASA envío a Marte un dispositivo móvil tele operado denominado Sojourner Rover, el cual estaba encaminado al envío de fotografías del ambiente del mencionado planeta. De igual manera, en ese período fiscal la compañía japonesa HONDA, dio a conocer el robot P3, el primer humanoide con la capacidad de imitar movimientos humanos.

Al siguiente año, en la Universidad Waseda de Japón, se da a conocer el WABIAN R-III, un autómata humanoide con destino investigativo acerca del movimiento del cuerpo

humano. En el año 1999, en el CMU, Zeglin planteó un nuevo diseño de robot con una pata denominado Bow Leg Hopper, el cual otorga la posibilidad de almacenar la energía potencial de la pata, partiendo del hecho de que es una articulación con un diseño curvo, el cual, al cumplir cada paso, realiza la compresión de un sistema de resorte; y de esta manera, resulta mínimo el consumo de energía, otorgando el ahorro significativo de la mencionada.

A su vez, Hollis et al. en el año 2006 desarrolló a Ballbot, un sistema holónimo cuyo desplazamiento es otorgado por una esfera situada en el área de debajo de la estructura; donde el control de la esfera es parecido al de un mouse de computadora, utilizando encoders a fin de delimitar cada posición de la misma; a pesar de esto, el estudio de esta tipología de autómatas con una esfera fue iniciado por Koshiyama y Yamafuji en el año 1991. En la actualidad, los robots tele operados Spirit Rover y Opportunity Rover, se hallan explorando el área del planeta Marte, con el objeto de ubicar mantos acuíferos.

2.3.2. Robot Móvil.

Calle (2007) citado por Ramírez y Reyes (2015) manifiesta que con el transcurso de los años, la robótica ha ido desarrollándose con los avances de la tecnología que se han venido ocasionando; escenario que ha originado el desarrollo de gran cantidad de aplicaciones en diversas áreas; aunque, a pesar de haber logrado un grado alto, en la actualidad todavía se debe esperar para la llegada de la revolución social de la robótica; convirtiéndose la totalidad de los autómatas en un componente adicional en el hogar, las organizaciones y la sociedad en general.

En la actualidad, los autómatas en su gran mayoría presentan costos exorbitantes, además de una función muy especializada, por lo que, en estos años, las investigaciones se dirigen a lograr que dichos autómatas sean más universales y baratos; otorgándole mayor acceso a la totalidad de individuos que deseen solucionar cierto inconveniente en sus labores habituales.

Comúnmente, los autómatas móviles presentan diminutos tamaños y un reducido nivel de control; adquiriendo un elevado nivel de dificultad al momento de ejecutar labores cooperativas; siendo determinado el comportamiento y la manera de operar de acuerdo al programa que se realiza en el mismo. Así pues, el software de los autómatas en la actualidad, se estructuran en tres niveles:

1. Sistema operativo.
2. Plataforma de desarrollo.
3. Aplicaciones concretas.

Ahora bien, la realización eficiente de las tareas del autómata depende de la construcción mecánica y de la programación; en cuanto a la precisión en su desplazamiento, está delimitado por su sistema mecánico; y la autonomía y la inteligencia, están presentes en el programa que dirige las acciones del microbot. (Ramírez & Reyes, 2015)

En el caso de un robot móvil independiente, el mismo se categoriza por presentar una unión inteligente entre las operaciones de percepción y acción, la cual precisa su actuación, otorgándole la posibilidad de alcanzar los objetivos proyectados dentro de ambientes con cierta indecisión. Así, el nivel de independencia depende en gran medida, de la capacidad del autómata para separar el entorno y transformar la información emanada en órdenes; de tal forma, que, aplicadas sobre los actuadores del sistema de locomoción, asegure la ejecución eficiente de su labor.

Es de esta forma, que las dos principales especificaciones que lo alejan de cualquier otra tipología de vehículo son las que se mencionan:

- **Percepción:** El autómata mencionado, tiene que poseer la capacidad de establecer la conexión con su ambiente laboral, a través del sistema sensorial a bordo. Dicha capacidad se resume en la síntesis de la cantidad de información provista por los sensores, a modo de producir mapas globales y locales del entorno; según los distintos niveles de control.

• **Razonamiento:** El mismo debe tener la capacidad de tomar la decisión de cuáles acciones son solicitadas en cada instante, partiendo del estado del robot y el de su ambiente, para lograr sus metas. En cuanto a la capacidad de razonamiento del autómata móvil, se convierte en la planificación de trayectorias globales seguras y en la habilidad para modificarlas en presencia de obstáculos inesperados (control local de trayectoria) para permitirle al autómata, el alcance de los objetivos confiados.

Un esquema básico general de la estructura de control de un robot móvil y las partes que componen la arquitectura general de control; es el siguiente:

- Generador Global de Trayectorias (GGT): Se trata del nivel jerárquico superior; que se encarga de tomar la decisión según la labor otorgada, las coordenadas del punto destino, de puntos intermedios en el camino y, cuando éste se encuentre atascado, redelinear el trayecto seleccionado. Es por ello, que es posible que los datos utilizados en este nivel, se generen fuera de línea (juicio previo del ambiente laboral); o en línea, basado en juicios ya definidos y empleando la información provista por el sistema sensorial (desconocimiento parcial o total del ambiente de trabajo); iniciando de la realización de mapas del ambiente (SLAM).

- Generador Local de Trayectorias (GLT): Representa el nivel jerárquico intermedio, el cual actúa en ocasiones como operador (piloto) del autómata móvil, previniendo los impedimentos del recorrido; llevando a cabo, correcciones en el trayecto y ajustando la rapidez del transporte según la maniobra ejecutada. Otorga la posibilidad de un control dinámico del autómata móvil, a la vez que mantiene al tanto al GGT acerca de los resultados del objetivo designado; y en la situación de carencia de conocimiento previo del ámbito de labores, produce información que es depositada en la memoria del GGT. Ahora bien, se encuentra directamente comunicado con el sistema sensorial, escenario que le permite asumir decisiones en línea, así como generar los valores de referencia para el Control Local del Sistema de Tracción y Dirección. Han sido desarrollados GLT tanto con algoritmos clásicos del tipo Maze-Search, como con el empleo de elementos de la Inteligencia Artificial que emulan la actuación del operador humano.

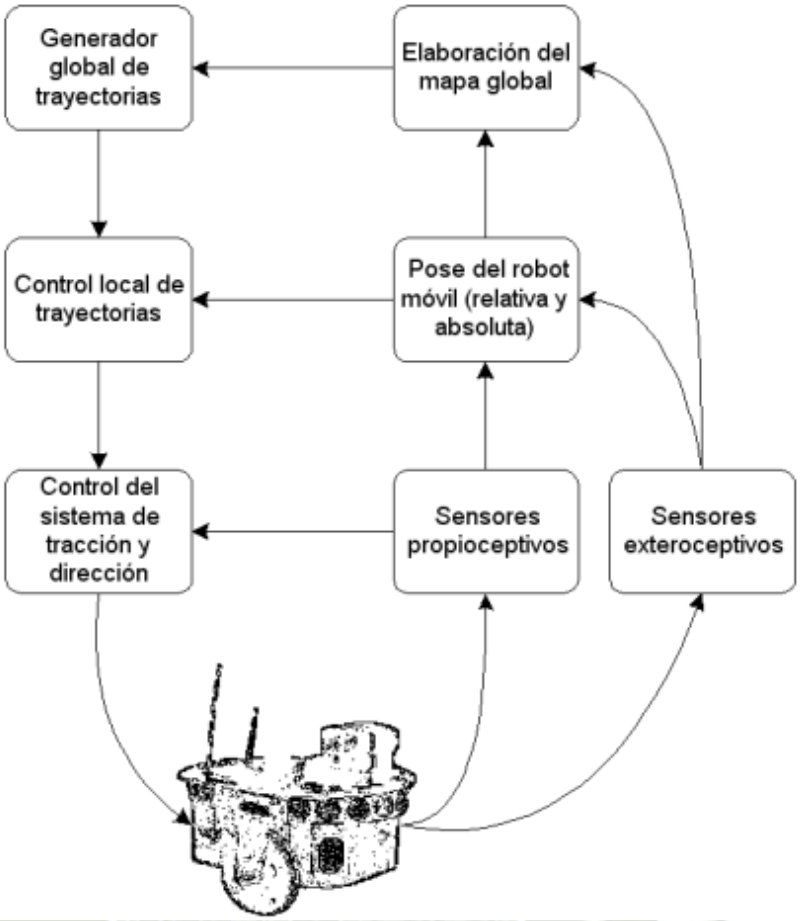


Figura 6. *Esquema general del sistema de control de un autómata móvil.*
Fuente: Bambino (2008).

- Control Local del Sistema de Tracción y dirección (CL): Actúa como el nivel jerárquico inferior; interpretando las referencias emitidas desde el GLT y provocando las acciones de control para que los motores de tracción y dirección laboren de manera coordinada; lo que permite que alcance el punto destino recorriendo trayectorias suaves, libres de oscilaciones y maniobras peligrosas para la carga. En el mencionado nivel, los controladores corresponden a los desarrollados en la teoría básica de control.

Ahora bien, la planificación de rutas en escenario inciertos es llevada a cabo mediante generadores locales de trayectorias que únicamente toman en consideración el ambiente cercano al autómata móvil para delimitar la dirección a seguir; los recorridos logrados no son idóneos. No obstante, en entornos donde se posee conocimiento, esta

planificación se lleva a cabo mediante los generadores globales de trayectorias, donde prioritariamente toman en consideración la totalidad de recorridos posibles, eligiendo aquel que posea un factor menor de costo; factor que se encontrará afectado por el tránsito, prioridades de circulación, densidad de obstáculos, entre otros, en las distintas trayectorias.

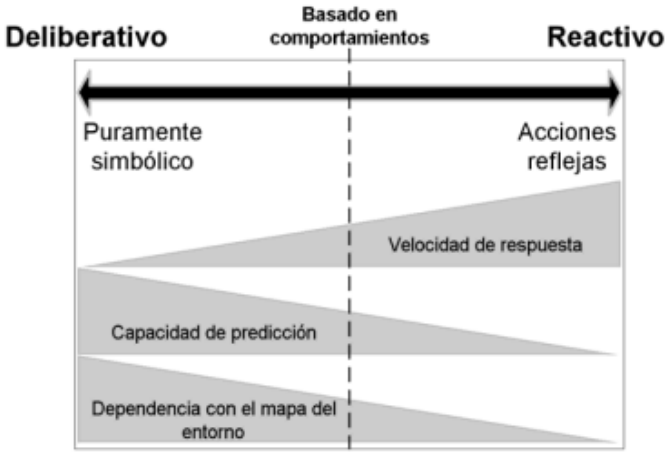


Figura 7. *Estrategias de control para autómatas móviles.*
Fuente: Bambino (2008).

En cuanto a las estrategias de control deliberativo, están sustentadas en una estrategia netamente simbólica; traduciéndose esto en el hecho de que la enseñanza entre el ambiente y su modelo en el autómata tiene que ser concisa a modo de que la actuación del mismo sea aquel que se espera. A pesar de ello, las estrategias deliberativas incorporan un estudio de sostenibilidad que permite asegurar a priori, según qué condiciones del ambiente el autómata logrará alcanzar sus objetivos; situación que conlleva a sistemas de inspección y de procesamiento de la información que invierten un gasto computacional relevante, que a su vez delimita su velocidad de respuesta. (Bambino, 2008)

Ahora bien, las estrategias de control reactivo están sustentadas en un esquema de acciones reflejas, lo que se interpreta como que el ambiente es visto como un estímulo que ocasiona una acción de control en base de la energía de éste. Entonces, la mencionada autonomía con el ambiente, unida a funciones de control simples, potencia

dicha tipología de estrategias partiendo de su alta velocidad de respuesta y su reducido gasto computacional.

2.3.3. Tipos de ambientes operativos para Autómatas móviles.

Siguiendo la línea del mencionado autor, en esta especificación es donde se encuentran establecidas más cantidad de limitaciones sobre el autómata móvil, agrupándose en la zona de trabajo y de acuerdo con los cuerpos que se encuentren ubicados en el ambiente. (Bambino, 2008)

Partiendo de ese supuesto, el ambiente del autómata podría ser interior o exterior. El primero, se encuentra definido por paredes y cielorrasos; con una iluminación básicamente artificial; mientras que el segundo, no se encuentra bien delimitado, donde la iluminación generalmente es natural.

Figura 8. *Robot de interior y robot de exterior.*
Fuente: Bambino (2008).

De acuerdo con los objetos que se encuentran ubicados en el ambiente, el mismo puede categorizarse como estructurado o no estructurado. Estructurado, se refiere al escenario donde los objetos que se ubican en él son estáticos, ya que no modifican su forma ni de posición y tienen especificaciones físicas específicas (forma, color, entre otras); las cuales permiten relacionarlos con figuras geométricas como prismas o cilindros, o

incluso realizar la diferenciación de unos objetos de otros (puertas abiertas, mesas de trabajo, etc.).

En el caso del ambiente no estructurado, se refiere en la situación donde la asociación entre los objetos del ambiente y ciertas especificaciones físicas no son posibles; debido a que cuando el ambiente es cambiante, éstos pueden resultar impredecibles (Bambino, 2008).

2.3.4. Tipos de Sistema de Locomoción.

El ambiente, viene a ser un condicionante al sistema de locomoción de un autómata; pudiéndose ser, según las particularidades del mismo:

1. Terrestre: Con patas, con ruedas y con cadenas.

Figura 9. *Ejemplos de Autómatas terrestres.*
Fuente: Bambino (2008).

2. Acuático: Flotante, submarino y aéreo.

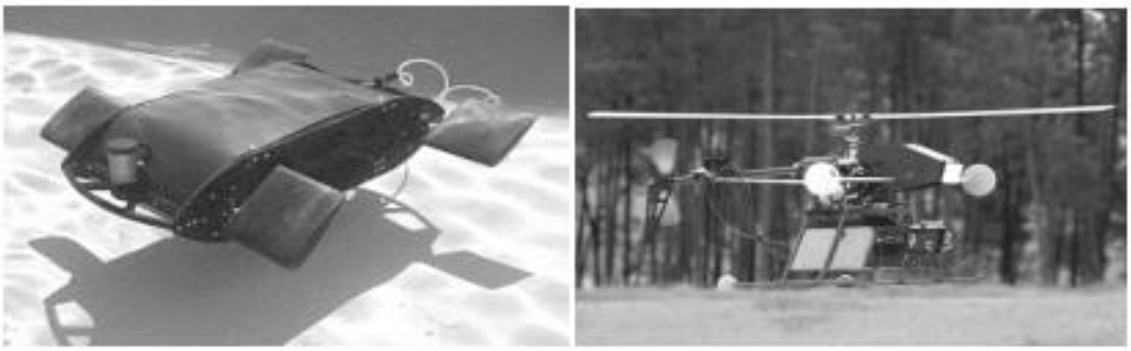


Figura 10. Ejemplos de Autómatas Acuáticos.
Fuente: Bambino (2008).

2.3.4.1. Tipologías de Ruedas.

Cuanto se trata de autómatas terrestres que presentan ruedas, su desplazamiento se destaca por dos aspectos: la categoría de rueda que presentan, y su disposición sobre una estructura mecánica; presumiendo el transcurso del desplazamiento, el plano de la rueda se dispone de manera vertical, rotando las mismas alrededor de su eje (horizontal), que posee una dirección con relación a la estructura, la cual puede resultar estática o cambiante.

Ahora bien, Bambino (2008), hace una diferenciación de ruedas, comentando que pueden ser: la convencional, donde el contacto entre ésta y el suelo se presume que cubre las expectativas de la rotación pura sin resbalamiento; por lo que la velocidad del punto de contacto tiene una igualdad a cero. En lo que respecta a la rueda sueca, es presumible que el contacto entre ésta y el terreno es disminuido a un punto único del plano.

Por otra parte, se pueden observar tres tipologías en cuanto a las ruedas convencionales:

1. **Rueda Fija:** Cuando el eje de la misma se encuentra fijado a la estructura del robot; comúnmente relacionada al sistema de tracción del mismo.

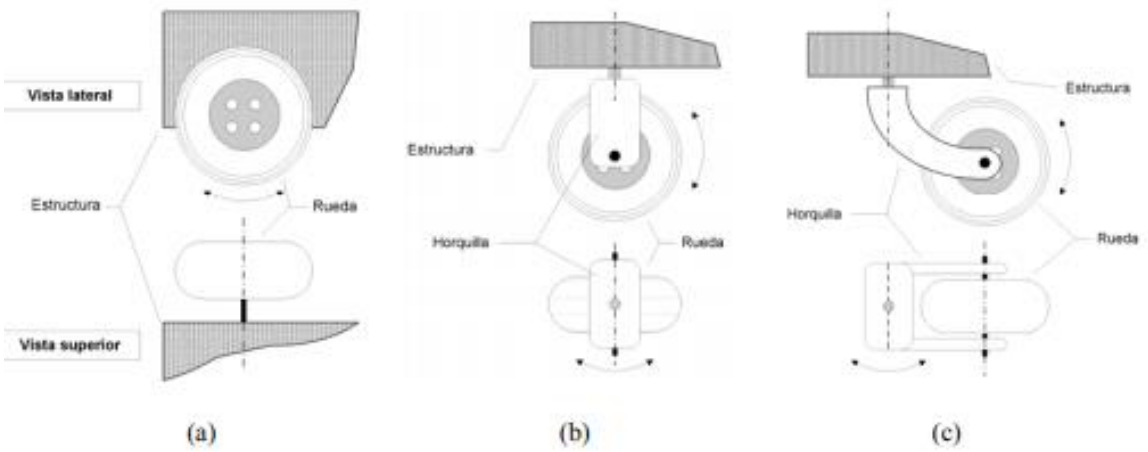


Figura 11. *Rueda Fija (a), Rueda orientable centrada (b) y rueda loca (c*).
Fuente: Bambino (2008).

2. **Rueda orientable centrada**: En esta tipología, el movimiento del plano de la rueda en referencia a la estructura representa una rotación alrededor de un eje vertical que pasa por medio del punto central de la rueda; por lo que asume funciones de dirección o como rueda tracción-dirección.
3. **Rueda orientable no centrada (rueda loca):** Se reconoce igualmente como rueda castor; la cual es orientable con referencia a la estructura, de manera que la rotación del plano de la rueda es alrededor de un eje vertical; que no atraviesa el punto central de la misma; por lo que su finalidad es otorgar estabilidad a la edificación mecánica del autómata.

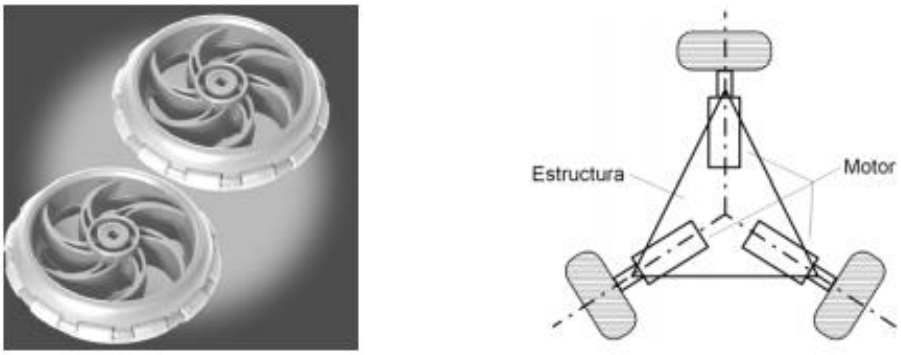


Figura 12. *Detalle de una rueda sueca y su disposición sobre la estructura mecánica del autómata.*

Fuente: Bambino (2008).

2.3.4.2. Disposición de las Ruedas.

Debido a la mezcla de distintas tipologías de ruedas, resulta posible dispone de una gran diversidad de autómatas móviles; haciéndose una distinción de acuerdo al nivel de maniobrabilidad.

a. Robot Omnidireccional.

Se caracterizan por una maniobrabilidad elevada en el plano, ya que pueden desplazarse en diferentes direcciones, aun sin tener que ser reorientados. En otros términos, el autómata es capaz de girar, avanzar o movilizarse lateralmente, debido a la rotación de cada rueda que posee. Ahora bien, las ventajas de un robot omnidireccional se ven disminuidas por la complejidad mecánica y/o electrónica requerida para la conservación de una buena coordinación entre las ruedas; y de esta manera, prevenir derivas en la pose del robot.

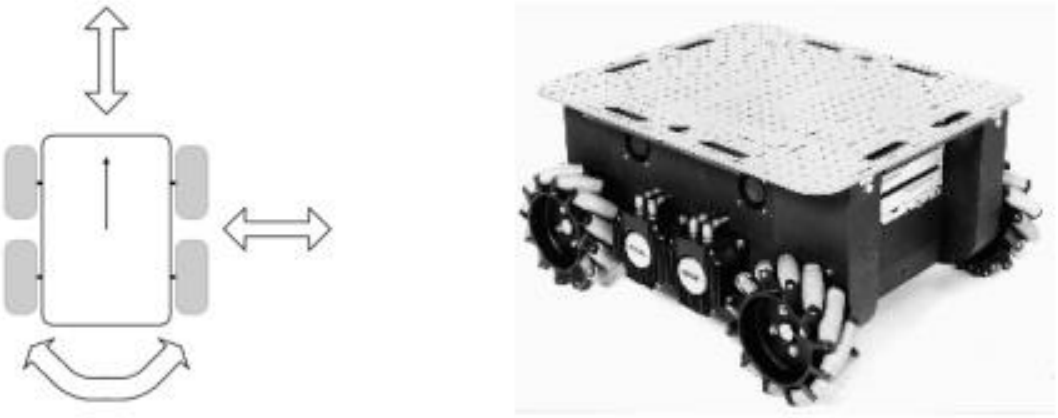


Figura 13. *Autómata Omnidireccional con ruedas suecas.*

Fuente: Bambino (2008).

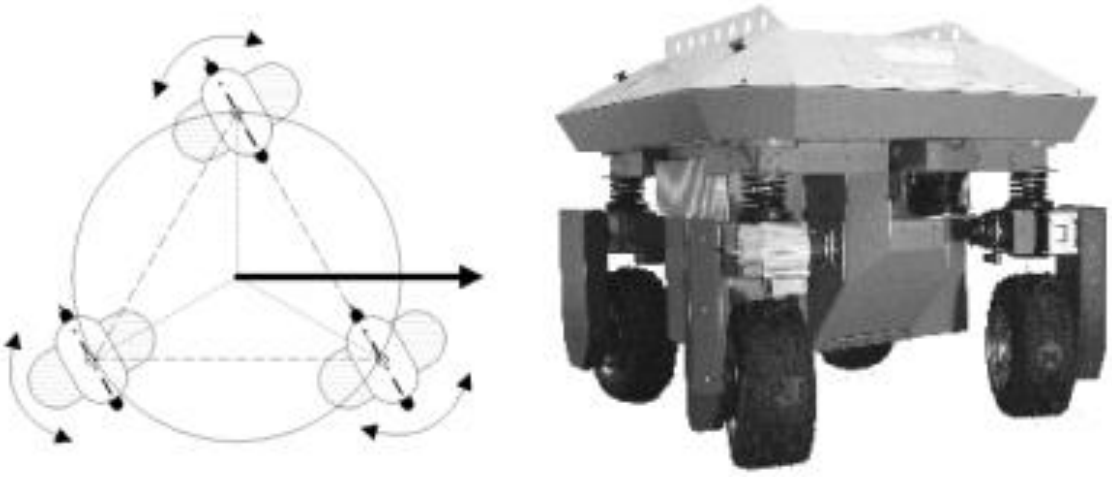


Figura 14.*Robot omnidireccional con ruedas orientables centradas.*

Fuente: Bambino (2008).

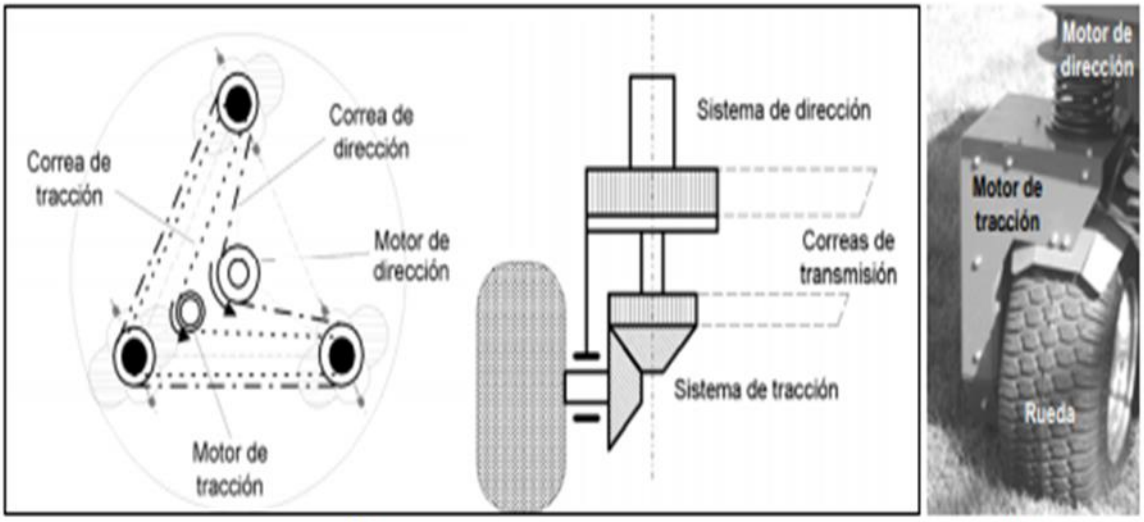


Figura 15.*Sincronismo entre el sistema de tracción y dirección con ruedas omnidireccionales (a) Mecánico y (b) Electrónico.*

Fuente: Bambino (2008).

b. Uniciclo.

Representa una estructura que consta de dos ruedas fijas convencionales sobre el mismo eje, controladas de manera independiente y una rueda loca que le confiere estabilidad (Bambino, 2008).

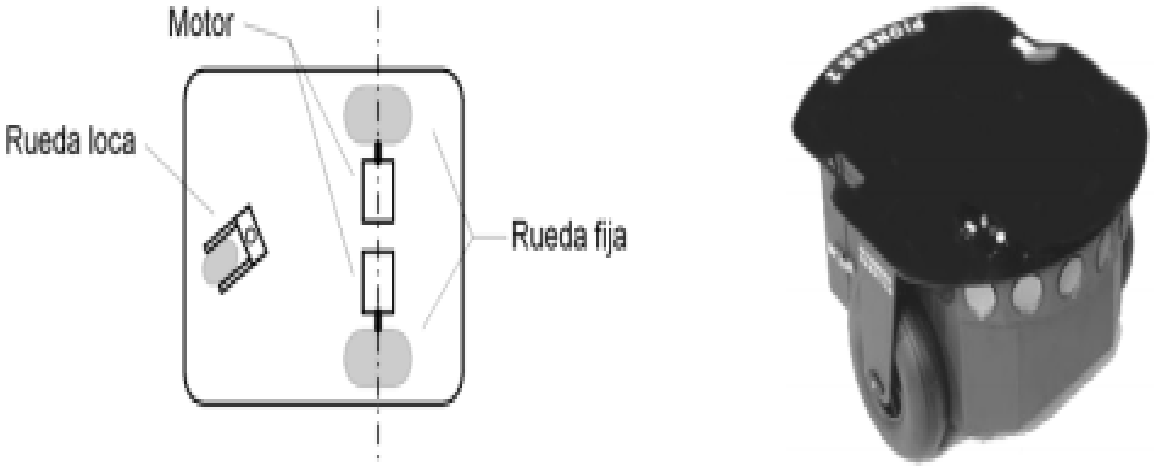


Figura 17. *Autómata Uniciclo.*

Fuente: Bambino (2008)**.**

c. Triciclo.

Se encuentra conformado por dos ruedas convencionales fijadas sobre un mismo eje, y una rueda convencional centrada orientable que concentra las funciones de tracción-dirección.

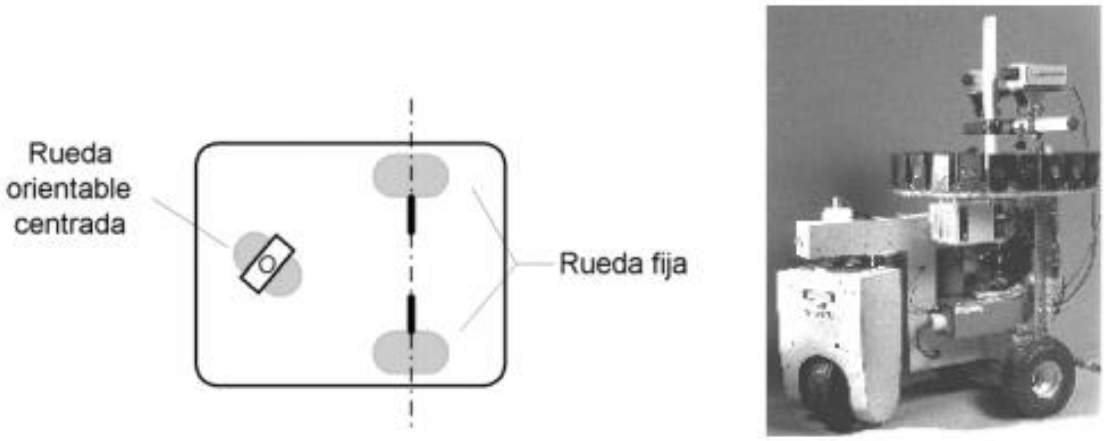


Figura 16. *Triciclo.*
Fuente: Bambino (2008).

d. Cuatriciclo.

Esta categoría, presenta una gran dificultad, y es que el centro de gravedad del vehículo se posiciona en ciertos momentos, en las limitantes de la superficie de equilibrio delimitada por tres ruedas; en las situaciones donde el vehículo se encuentra en desplazamiento; escenario que genera una pérdida de tracción en el vehículo, convirtiéndose en una fuente de error cuando se quiere estimar la posición del autómata.

2.3.5. Tracción y dirección del Robot Móvil.

Bambino (2008) expresa, que el sistema de tracción y dirección de un autómata, posee conexión con las ruedas adoptadas, además de con los algoritmos de control local de los motores y la mecánica relacionada a éstos. Es así, que se presentan tres sistemas primarios, desde donde es posible alcanzar distintas configuraciones:

2.3.5.1. Tracción y dirección en ejes independientes.

En las ruedas traseras, se lleva la tracción, mientras que el control de dirección en las ruedas delanteras; o viceversa. En cuanto al control de la dirección, la precisión en la misma tiene dependencia de la adherencia de las ruedas correspondientes, debido a la masa despreciable de dichas ruedas con relación al resto de la estructura. Igualmente, presenta un radio de giro lo suficientemente alto en correlación a otros sistemas, lo cual ocasiona que no se logren alcanzar los cambios de dirección muy cerrados.

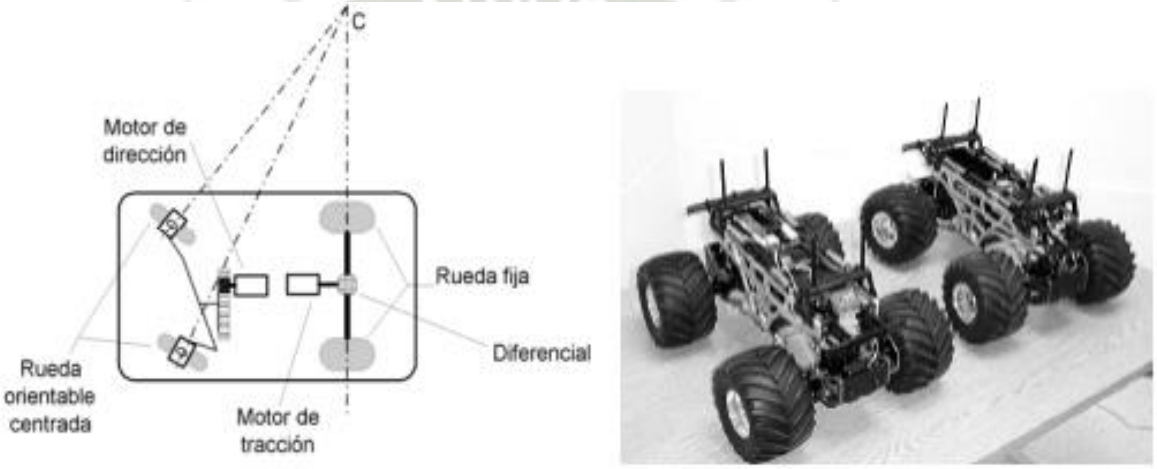


Figura 17. *Sistema de tracción y dirección en ejes independientes.*
Fuente: Bambino (2008).

2.3.5.2. Tracción y dirección en un mismo eje.

Corresponde a un modelo de construcción con poca complejidad, que permite radios de giro del orden de la dimensión del vehículo; aunque su desventaja radica en que los motores requieren de particularidades iguales, a modo de que su control sea simple.

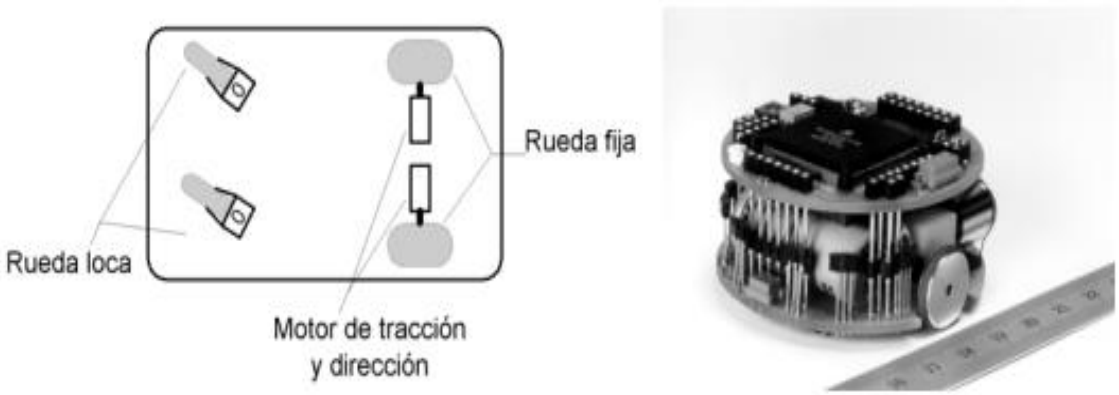


Figura 18. *Sistema de tracción y dirección sobre un mismo eje.*

Fuente: Bambino (2008).

2.3.5.3. Tracción y dirección sobre todos los ejes.

En esta configuración, se requiere de un sistema odométrico con elevada complejidad, debido al desconocimiento en los radios de giro relacionado al mismo.

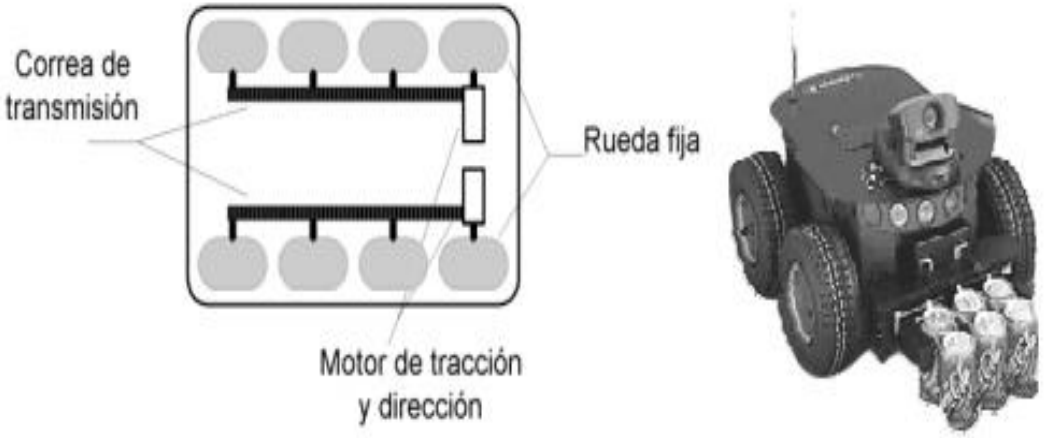


Figura 19.*Sistema de tracción y dirección sobre todos los ejes.*

Fuente: Bambino (2008).

2.3.6. Estructura de un Robot Móvil.

La estructura de un autómata móvil guarda mucha similitud a la estructura de un ser vivo, ya que comúnmente se encuentran destinados a realizar la simulación de la actuación de personas y animales, incluso con un grado de eficiencia parecida.

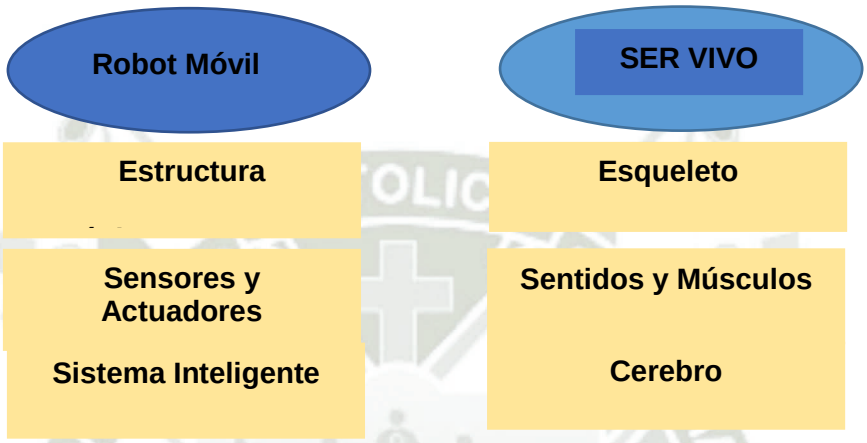


Figura 20. *Similitudes entre un Robot y Ser Vivo.*
Fuente: Ramírez & Reyes (2015).

Partiendo de la estructura del robot móvil expuesta, se aprecia que se encuentra integrado por distintas estructuras, como:

- Estructura Mecánica: estructura con ruedas, patas y orugas.
- Actuadores: Motores, luces, brazos, ruedas, o cualquier elemento que le haga posible relacionarse con el ambiente.
- Sensores: Sonar, láser, cámaras y cualquier otro dispositivo que otorgue datos del ambiente.
- Inteligencia: Métodos, algoritmos, entre otros; los cuales permitirán relacionarse con el ambiente, a partir de los registros de datos de los sensores.

2.3.7. Grados de Libertad de un Robot Móvil.

Para calle (2007) mencionado por Ramírez & Reyes (2015), los grados de libertad de un autómata vienen a ser cada uno de los movimientos de desplazamiento y rotación que puede llevar a cabo el robot. Así pues, un cuerpo que se desplaza en dos dimensiones, posee 3 GDL (una rotación y 2 traslaciones); mientras que un cuerpo que se desplaza en tres dimensiones, posee 6 GDL (3 rotaciones y 2 traslaciones).

2.3.8. Clasificación de los Robot Móviles.

Ramírez (2017) citado por Pérez (2019), el desplazamiento de los robots móviles, se ejecuta a través del control de mandos, o bien, al guiarse con datos captados del ambiente, a través de sensores; presentando un nivel más elevado de autonomía, provecho y rapidez que en el caso de un dispositivo permanente. Ahora bien, los sensores y actuadores, se refieren a módulos de comunicación con el hardware del autómata; en el caso de que sea requerido tener acceso a información respectiva a los sensores, o en su defecto enviar comandos a los actuadores, lo realizan por medio de éstos.

Por su parte, Torres (2009) mencionado por el anterior autor, dice que los autómatas móviles terrestres pueden categorizarse de acuerdo a su estructura, así como la configuración de sus componentes; teniendo presente que dichos dispositivos manejan una capacidad para movilizarse gracias a su sistema locomotor, el cual permite poseer diversidad de sistemas de locomoción, según la superficie sobre la que se va a desplazar; ya que en función de la misma, es posible adaptar ruedas, orugas, patas articuladas o mecanismos híbridos. (Pérez, 2019)

Por ello, se pueden mencionar la siguiente tipología:

a. Robot Móvil con locomoción por ruedas

Esta categoría de locomoción, resulta la más habitual y con mayor estabilidad, partiendo del hecho de que en todo momento están situadas en el piso; donde la geometría de las ruedas depende de la tipología del área donde se deslizarán.

En la mencionada categoría, es posible ubicar diversas configuraciones según la posición y el número de ruedas del autómata. Al respecto, inicialmente se observa la configuración Ackerman, la cual se integra por dos ruedas traseras con tracción y dos ruedas delanteras de dirección; otra tipología de configuración es de tres ruedas, siendo reconocida como triciclo, el cual consta de dos ruedas motorizadas y una omnidireccional, la cual no tiene dependencia de un motor, ya que la misma gira de manera libre según la velocidad aplicada por el robot.

Figura 21.*Locomoción por ruedas.*
Fuente: Pérez (2019).

b. Robot móvil con locomoción por patas

Se tiene una diversidad de tipologías, inmersos en la clasificación de los robots móviles donde se utiliza traslado por patas; donde algunos de estos, son denominados bípedos que buscan parecerse a los movimientos ejecutados durante el proceso de caminar de un ser humano; y otros, son llamados zoomórficos, los cuales intentan parecerse a la biología de ciertos animales.

En la mayoría de los casos, la locomoción por patas se encuentra sustentada en la imitación de movimientos ejecutados por seres biológicos, ya sean humanos o animales.

c. Robot Móvil con locomoción por orugas

Esta sistematología es considerada como una opción a las ruedas y patas, al tratarse de sitios compactos o áreas irregulares, empleando pistas de desplazamiento, proporcionando una zona mayor de contacto con el suelo; otorgando este sistema, una tracción más óptima que las ruedas, y una estabilidad más elevada que las patas.

Figura 22. *Locomoción por orugas.*

Fuente: Pérez (2019).

Por su parte, Bambino (2008), realiza una clasificación global de los autómatas móviles en:

1. **Industriales.** Tienen la más alta difusión en funciones de tipo financiera; conformados por una estructura mecánica articulada la cual tiene movimiento al adoptar distintas configuraciones, por las indicaciones obtenidas de un equipo de control, apoyado por lo general en un microprocesador. Tienen la posibilidad de mover cargas pesadas a grandes velocidades, y con una elevada exactitud

Figura 23.*Robot Industrial PUMA (Unimation).*
Fuente: Bambino (2008).

2. **Médicos.** Éstos autónomos pueden ser de cooperación o de rehabilitación, siendo creados como prótesis inteligentes; las cuales se procura que posean la apariencia de la extremidad humana que corresponda, para llevar a cabo las funciones de la mencionada extremidad; donde las señales de mando resultan de señales nerviosas o musculares. Así mismo, en esta tipología se encuentran los autómatas que han sido creados como asistentes en actividades quirúrgicas de alta precisión o dificultad.

3. **Móviles:** Se refieren a dispositivos de transporte automático; en otras palabras, una plataforma mecánica equipada de un sistema de locomoción con la capacidad de navegar mediante un ambiente de labores específico, dotado de cierto nivel de autonomía para su deslizamiento llevando cargas. Son múltiples sus aplicaciones, teniendo conexión con actividades que comunmente son peligrosas o nocivas para la salud de la persona. (Bambino, 2008)

2.3.9. Sensores.

Un sensor es un dispositivo “sensible” que detecta cambios en el entorno y responde a una salida en el sistema principal. Es decir, que transforma un fenómeno físico en un voltaje analógico medible.

- Módulo sensor HCSR-04, pequeño sensor ultrasónico de buena precisión, bajo costo y bajo consumo energético, permite medir distancia mediante ultrasonido, para determinar la distancia a la que se encuentra un objeto de 2 a 450cm. Bastante utilizado en proyectos como robots laberinto o sumo y sistemas de automatización por medición de distancia. Posee dos transductores, un emisor y un receptor piezoeléctricos. Donde el emisor emite 8 impulsos de ultrasonido (40KHz) luego de recibir la orden en el pin TRIG, el sonido luego de viajar rebota y es detectado por el receptor piezoeléctrico, luego el pin ECO cambia a Alto (5V) por un tiempo igual al que demora en la onda desde ser emitida hasta ser detectada en el rebote, el tiempo del pulso ECO se mide con el microcontrolador y calcula la distancia del objeto. No le afecta la luz, ni el color negro, aunque con ciertos materiales blandos acústicamente puede ser difícil que los detecte. (Naylamp Mechatronics, s.f.)
- MPX10DP, Sensor de presión de silicio descompensado. Transductor sin compensación de temperatura con un rango de medición de 0 a 10 KPa (0-1,45 PSI) que proporciona una salida de tensión muy precisa y lineal, directamente proporcional a la diferencia de presión aplicada. Este sensor emplea galgas extensiométricas de silicio (Si) con una sensibilidad de 3,5 mV/KPa ante una alimentación típica de 3 voltios. (Omega, 2021)
- Módulo sensor AMG8833, este sensor de Panasonic es una matriz 8x8 de sensores térmicos IR (para medir temperatura). Cuando se conecta a su microcontrolador (Raspberry Pi) devolverá una matriz de 64 lecturas

individuales de temperatura infrarroja sobre I2C. Medirá temperaturas que oscilan entre 0°C y 80°C con una precisión de +- 2.5°C. Como sofisticadas cámaras térmicas, pero lo suficientemente compactas y simples para una fácil integración. (Robokits India, s.f.)

- Módulo sensor MAX30100, permite medir el porcentaje de saturación de oxígeno de la hemoglobina (SaO2) en la sangre utilizando un circuito fotoeléctrico (pulsioximetría no invasiva). El dispositivo integra los emisores de luz y el sensor que mide la cantidad de luz reflejada a través del dedo de la persona, siendo que la sangre oxigenada absorbe más luz infrarroja, y la poco oxigenada más luz roja, a lo cual el sensor detecta la luz para calcular la concentración de oxígeno. (Naylamp mechatronics, s.f.)
- Sensor shield Raspberry Pi, dispositivo de rendimiento biométrico y aplicaciones médicas, puede realizar un monitoreo corporal a través de 9 sensores distintos, lo que permite hacer una lectura del paciente en tiempo real para luego analizarlos en el diagnóstico médico. La información recopilada puede ser enviada de forma inalámbrica. (Cooking-hacks, s.f.)

2.3.10. Raspberry Pi 4, 4GB RAM

Básicamente es un ordenador adaptado a necesidades de programación muy básicas, de placa reducida, bajo coste, sistema operativo abierto Raspbian. Inicialmente fue diseñado para la enseñanza de informática en escuelas, sin embargo, resalta por sus capacidades emergentes, las cuales gracias a los usuarios son cada vez más variadas. Pudiendo usarse como un ordenador, hasta como controlador para robots, entre miles de otras aplicaciones.

Esta versión es más potente y moderna, ya que tiene un tamaño práctico, similar a una tarjeta de crédito (8.56cm x 5.65 cm) y un peso de 45 gramos. Es un ordenador de placa única (SBC), tiene las funciones básicas de una computadora, navegación por internet, vídeo, documentos, juegos, programaciones. Su funcionamiento es similar al

Arduino, lo que permite usar pines GPIO y comunicación tipo I2C y SPI. Sin embargo, su practicidad viene siendo aprovechada y explorada en proyectos sobre electrónica y robótica, ya que pueden ensamblarse sensores y actuadores, que lo hacen sumamente útil en procesar imágenes, vídeos, cámaras y cálculos matemáticos complejos. Su componente principal es System on Chip (SoC) Broadcom BCM2711, que cuenta con un chip regulador de voltaje, que aumenta la eficiencia de energía, además cuenta con un procesador ARMv8 de 4 núcleos de 64 bits, con una velocidad de 1.5 GHz, GPU Broadcom Video Core VI y Memoria RAM de 4GB LPDDR4, que le permite funcionar con ARM, GNU/Linux, aunque Raspberry diseño su propio sistema Raspberry Pi OS, y funciona también con Sanppy Ubuntu Core y Microsoft Windows 10 IoT. Tiene 2 puertos USB comunes y 2 puertos USB 3.0 de mayor velocidad, conexión micro-HDMI (type-D) para 2 monitores, Wifi doble banda, Bluetooth 5.0 BLE, Ethernet Gigabit mejorado, entre otras características que lo hacen uno de las potentes del mercado actual. (Santana, 2019)

2.3.11. COVID 19.

2.3.11.1. Definición de COVID 19.

Una gran cantidad de expertos en el área de salud, consideran que el nuevo virus de coronavirus pudo haberse originado en murciélagos o pangolines; generándose la transmisión inicial al ser humano en la ciudad de Wuhan, China.

Los coronavirus, vienen a ser una agrupación de virus que tienen la posibilidad de ocasionar enfermedades tanto en animales como en seres humanos. Aproximadamente, el 80% de los individuos con COVID 19 llega a mejorarse sin la necesidad de un tratamiento especializado, ya que solo experimentan síntomas leves, que presentan parecido a los síntomas de una gripe; aun cuando, 1 de cada 6 seres humanos suele manifestar síntomas graves como problemas respiratorios.

Ahora bien, el COVID 19 se ha propagado de manera rápida por muchos lugares del mundo; razón por la cual la Organización Mundial para la Salud lo declaró el 11 de marzo del año 2020 como una pandemia mundial (Kandola, 2020).

2.3.11.2. Proceso de Transmisión.

El SARS-COV-2 o COVID-19, se transmite de individuo a individuo, a través de comunidades que presentan cercanía, aproximadamente a menos de metro y medio de persona a persona; por lo que cuando un individuo contagiado con COVD-19 exhala o tose, expulsa diminutas gotas que contiene el virus, las cuales pueden ingresar por la boca o nariz de otra persona causándole así la infección. Asimismo, las gotas que contienen el virus es posible que caigan encima de superficies u objetos cercanos, ocasionando que otra persona se contagie al tocar dichas superficies contaminadas.

Este padecimiento resulta más contagioso, en el momento en que los síntomas se encuentran en su apogeo, aunque existe la posibilidad de que una persona asintomática (que no manifiesta síntomas) pueda propagar el virus.

2.3.11.3. Síntomas del COVID-19.

Continuando con las ideas de Kandola (2020), los síntomas del padecimiento pueden manifestarse de 2 a 14 días luego de la exposición al virus; siendo los más sobresalientes:

- Tos persistente
- Fiebre
- Dolor de Cabeza
- Dificultad para respirar o falta de aire
- Dolor en el pecho
- Dolores en los músculos
- Escalofríos
- Dolor de garganta
- Dolor en el pecho

A este respecto, Mayo Clinic (2021) indica que se han reportado otros síntomas menos comunes, como erupción en la piel, náuseas, vómitos, y diarrea; destacando que los infantes manifiestan síntomas parecidos a los de los adultos, aunque por lo común manifiestan un padecimiento leve.

En lo que refiere al nivel de gravedad de los síntomas de la COVID-19, son manifestables desde muy leve a muy extrema; ya que ciertas personas manifiestan unos pocos síntomas, mientras que, en cambio, otras son capaces de no presentar alguno.

Al respecto, algunas afecciones de salud que ocasionan un incremento en el peligro de padecer de gravedad con la COVID-19, incluyen:

- Padecimientos cardíacos agudos, como insuficiencia cardíaca, enfermedades de las arterias coronarias, o miocardiopatía.
- Cáncer.
- Enfermedad pulmonar obstructiva crónica (EPOC).
- Diabetes tipo 2.
- Sobrepeso u obesidad grave.
- Fumar.
- Enfermedad renal crónica.
- Enfermedad de células falciformes.
- Sistema inmunológico débil a razón de trasplante de órganos sólidos.

Por otro lado, el peligro de padecer la enfermedad con mayor gravedad, es posible que se incremente a razón de otras afecciones, como son:

- Asma.
- Enfermedad hepática.
- Obesidad.
- Enfermedades pulmonares crónicas, como fibrosis quística o fibrosis pulmonar.
- Afecciones del cerebro y del sistema nervioso.
- Sistema inmunitario debilitado por trasplante de médula ósea, VIH, o algunos medicamentos.
- Diabetes tipo 1.
- Presión arterial alta.

2.3.11.4. Medidas de Prevención.

Ferré (2020) manifiesta que existen una serie de medidas que ayudan a prevenir el contagio del COVID-19. Entre ellas se mencionan:

- Realizar el lavado de manos con regularidad (mínimo por un lapso de 20 segundos) o con soluciones alcohólicas; en especial, luego de tener contacto con personas enfermas o con el entorno.
- Prescindir tocarse los ojos, la nariz y la boca con las manos sucias.
- Evitar el contacto con personas que manifiesten síntomas de afección respiratoria, como tos o estornudos.
- No compartir comida, utensilios (cubiertos, vasos) y otros utensilios, sin una limpieza previa adecuada.
- Mantener un distanciamiento de dos metros aproximadamente, ante aquellos individuos que manifiestan síntomas.
- Cubrirse la boca y la nariz con pañuelos desechables o con la cara interna del codo, al instante de toser o estornudar; procediendo a lavarse las manos inmediatamente.
- Lavar y desinfectar con regularidad los objetos y las superficies.

2.3.11.5. Factores de Riesgo del COVID.

Existen ciertos factores que ocasionan peligro de entrar en contacto con la enfermedad, aun cuando otros factores causan afectación en la evolución de un padecimiento grave; esto dependerá de la lejanía de la propagación del virus en el ámbito situacional del individuo. (Kandola, 2020)

Al respecto, la Organización Mundial para la Salud mantiene, que el peligro de desarrollar la enfermedad continúa con un bajo nivel para la mayor parte de las personas, aun cuando esta situación se encuentra variando en cuanto que el virus se propague.

Así pues, el riesgo resulta más elevado para aquellas personas que mantengan contacto cercano con individuos que padezcan de COVID 19; o también, su propagación es más elevada en ciertas áreas, aquellas que por lo general se encuentren más pobladas. Finalmente, el riesgo más elevado lo poseen los adultos mayores y los individuos que presenten dificultades de salud crónica, como el caso de presión elevada, enfermedad cardíaca o pulmonar, o diabetes.

2.3.11.6. Complicaciones del COVID-19.

A pesar de que la mayor parte de las personas que padecen el virus del COVID-19, manifiestan síntomas entre leves y moderados, es posible que el padecimiento conlleve a dificultades médicas graves y, en ciertas personas, ocasionar el fallecimiento. En el caso de los adultos mayores, o aquellos individuos con afecciones crónicas, se encuentran expuestos a un peligro más elevado de contagiarse con el mencionado virus. A tal respecto, entre las complicaciones que pueden generarse, se encuentran:

- Neumonía y dificultades al respirar.
- Insuficiencia de varios órganos.
- Dificultades cardíacas.
- Afección pulmonar que ocasiona que un reducido volumen de oxígeno circule por medio del torrente sanguíneo a los órganos (síndrome de dificultad respiratoria aguda).
- Coágulos sanguíneos.
- Lesión renal aguda.
- Infecciones virales y bacterianas adicionales.

CAPÍTULO III

DESARROLLO DE INGENIERÍA DE DETALLE

3.1. Diseño Metodológico

3.1.1. Método de diseño

3.1.1.1. Metodología Top Down

La metodología Top-Down consiste comenzar con una idea con alto nivel de abstracción e ir incrementando el nivel de detalle a medida que se vaya avanzado en el diseño, esto se puede entender en la cuestión de fragmentar el problema en una serie de subproblemas de menor envergadura o más tratables, que bien podrían ser divididos en grupos aún más pequeños, y así sucesivamente, permitiéndonos obtener problemas que puedan ser resueltos de forma más sencilla y manejables. (OAS;, s.f.)

Usar esta metodología nos proporciona una manera de razonar enfocada en la búsqueda de las soluciones a los problemas de índole menor, y seguidamente relacionarlos con el objetivo de darle solución el problema general y principal, el cual fue desglosado en subproblemas. (OAS;, s.f.)

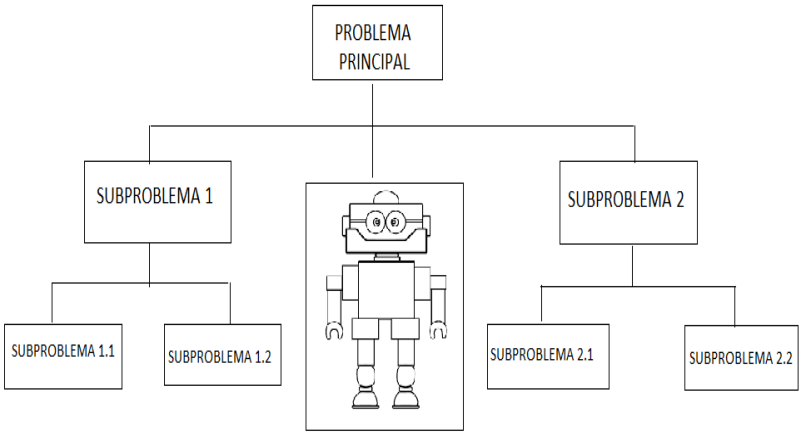


Figura 24. *Ejemplo de Diseño de Top Down.*

Fuente: Elaboración Propia.

3.1.2. Criterios de diseño

3.1.2.1. Requisitos y especificaciones del cliente

Tabla 1.

Requisitos y especificaciones del cliente (hospital/clínica/médico/paciente)

No	Requerimiento
1	Interactuar y obtener datos del paciente a distancia.
2	Trasladarse a otros ambientes para llevar hasta un 1kg de material médico o incluso alimentos.
3	Facilidad de mantenimiento
4	Durabilidad (3 años a más)
5	Economía (máximo entre S/. 3000 – S/. 4000)
6	De uso sencillo
7	Fácil interacción con el usuario
8	Trasmitir confianza
9	No debe interrumpir ni dañar a las personas
10	Debe ser útil

Fuente. Elaboración propia.

3.1.2.2. Requerimientos de función

Tabla 2.

Requerimientos de acuerdo a la función.

No	Requerimiento
1	Medir presión arterial, oxigenación (o saturación de oxígeno), ritmo cardíaco y temperatura.
2	Llevar material médico de un ambiente a otro.
3	Movimiento autónomo programado y remoto WIFI
4	No debe chocar
5	Software capaz de ensamblar el código para todos los procesos adecuadamente.

6	Interacción remota con el paciente
7	Entregar datos sensados al personal de salud
8	Manejo intuitivo
9	Seguridad industrial

Fuente. Elaboración propia.

3.1.2.3. Requerimientos estructurales

Tabla 3.

Requerimientos estructurales

No	Requerimiento
1	Base del Robot
2	Espacio adicional en la base para llevar material médico
3	Sistema de movimiento
4	Sistema de comunicación y control remoto
5	Suministro de energía
6	Interfaz humano
7	Sistema sensores
8	Controladores

Fuente. Elaboración propia

3.1.3. Diagrama de las conexiones del circuito.

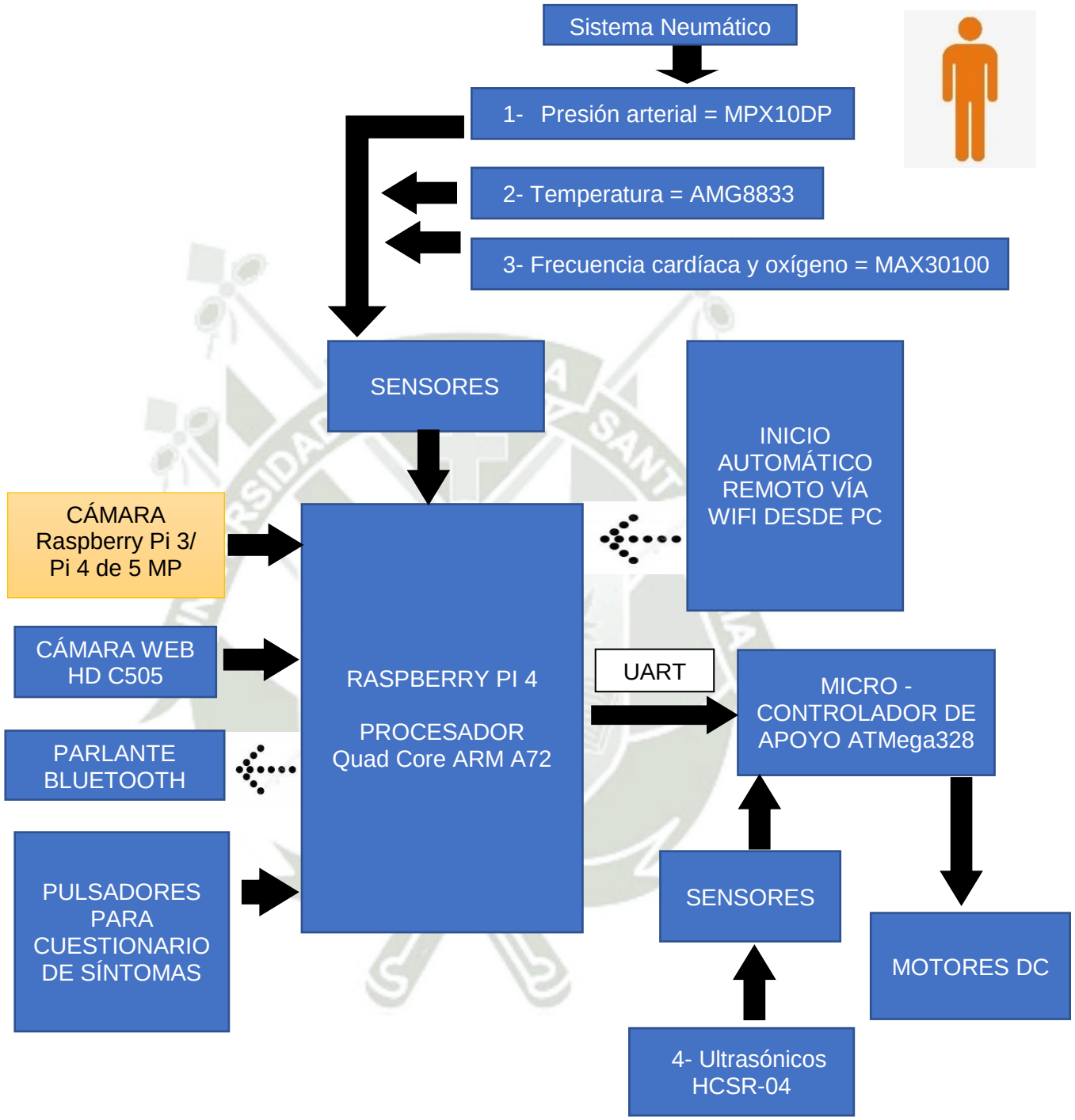


Figura 25. *Diagrama de Bloques General correspondiente a la composición del Robot Móvil.*
Fuente: Elaboración propia.
Flechas negras: conexiones alámbricas, Flechas punteadas: conexiones inalámbricas,

El robot móvil es controlado bajo la tarjeta Raspberry Pi 4, en la cual ejecuta el sistema operativo Raspberry OS de 32 bits, mismo que tiene preinstalado el intérprete de Python 3, lenguaje en el cual se desarrolló el programa principal. Además, para efectuar tareas que demandan mayor velocidad, se integra una tarjeta que incorpora el microcontrolador ATMega328.

La tarjeta Raspberry Pi 4 se encarga de ejecutar el programa principal, en el cual se llevan a cabo las siguientes tareas:

- Establecer comunicación remota vía wifi con el PC del médico o personal de salud (bloque INICIO AUTOMÁTICO REMOTO VÍA WIFI DESDE PC).
- Obtener captura de vídeo mediante una cámara Raspberry (bloque CÁMARA Raspberry Pi 3/ Pi 4 de 5 MP), para detectar y seguir el color AZUL de los conos, que representan la ruta hacia la cama del paciente.
- Obtener captura de vídeo mediante una cámara web (bloque CÁMARA WEB HD C505), para detectar si el paciente se encuentra en su cama y evaluarlo.
- Realizar un cuestionario de preguntas por medio del parlante, vía bluetooth, para conocer los síntomas del paciente (bloque PARLANTE BLUETOOTH).
- Recibir respuestas de cuestionario de síntomas por medio de dos pulsadores (bloque PULSADORES PARA CUESTIONARIO DE SÍNTOMAS).
- Recibir señales de sensores de toma de signos vitales (bloques SENSORES = TEMPERATURA, FRECUENCIA CARDÍACA Y OXÍGENO).
- Controlar un motor de bomba de aire y electroválvula, para toma de presión arterial (bloques SENSORES = PRESIÓN ARTERIAL, SISTEMA NEUMÁTICO).
- Mediante comunicación UART, enviar comandos al microcontrolador ATMega328 para activar/desactivar los motores.

Las funciones del microcontrolador ATMega328 son la de activar/desactivar los motores DC incorporados a las ruedas del robot (bloque MOTORES DC), y de detectar obstáculos y evadirlos al momento de estar a menos de 20 cm cerca de uno (bloque SENSORES = ULTRASÓNICOS). El funcionamiento de dicho microcontrolador está sujeto a los comandos recibidos por la tarjeta Raspberry Pi 4.

3.1.4. Descripción de funcionamiento

Previamente hay que señalar que se tuvo la idea inicial de desarrollar un robot móvil que sustituya a un enfermero asistente de tamaño ideal, para evitar contagios durante la pandemia COVID 19, así como se hizo en países como Alemania, Japón o China.

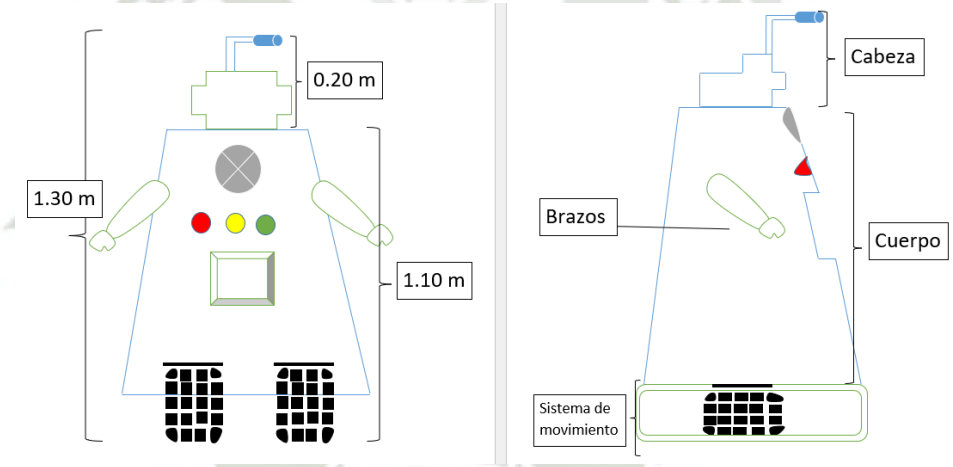


Figura 26. *Descripción del funcionamiento del robot.*

Fuente: Elaboración propia.

Como se ve en la figura anterior, este robot estaría compuesto por 4 partes, las cuales llamaremos cabeza, brazos, cuerpo y sistema de movimiento.

En la cabeza se encontraría la circuitería principal y cámara térmica. En los brazos están los sensores de oxigenación y ritmo cardíaco, y presión arterial. El cuerpo le da la estabilidad necesaria y tiene una cavidad para llevar material médico. El sistema de movimiento lo desplaza sobre cualquier superficie que pueda encontrar en el hospital.

Sin embargo, luego de hacer varias modificaciones hasta culminar la construcción del robot por completo, y manteniendo la idea inicial, se logra desarrollar el prototipo del robot móvil de la siguiente manera:

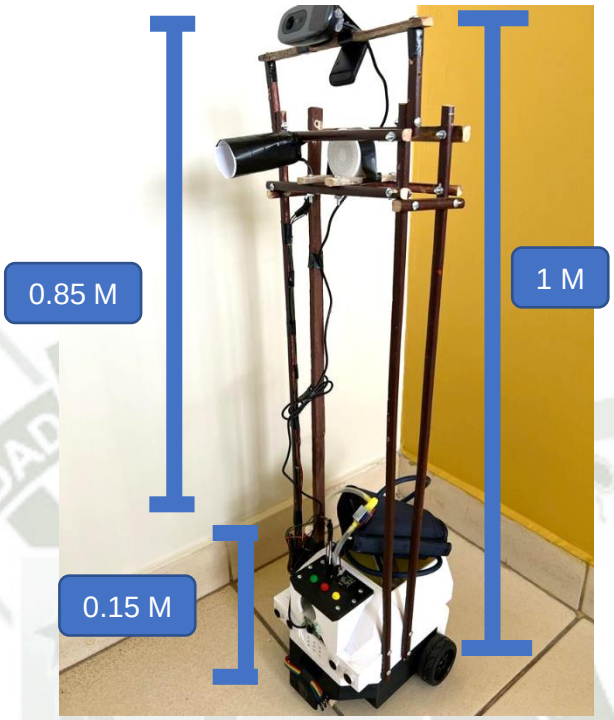


Figura 27. *Prototipo de robot móvil.*

Fuente: Elaboración propia.

Éste desarrolla las funciones principales de la idea inicial para evaluación del paciente: cuestionario de síntomas y toma de signos vitales (temperatura, la presión arterial, oxigenación, ritmo cardíaco, temperatura). Desplazamiento con 2 cámaras para encontrar al paciente y llevar material médico (como se ve en la siguiente figura). Y así mismo, de manera remota, enviará información de su estado de salud.

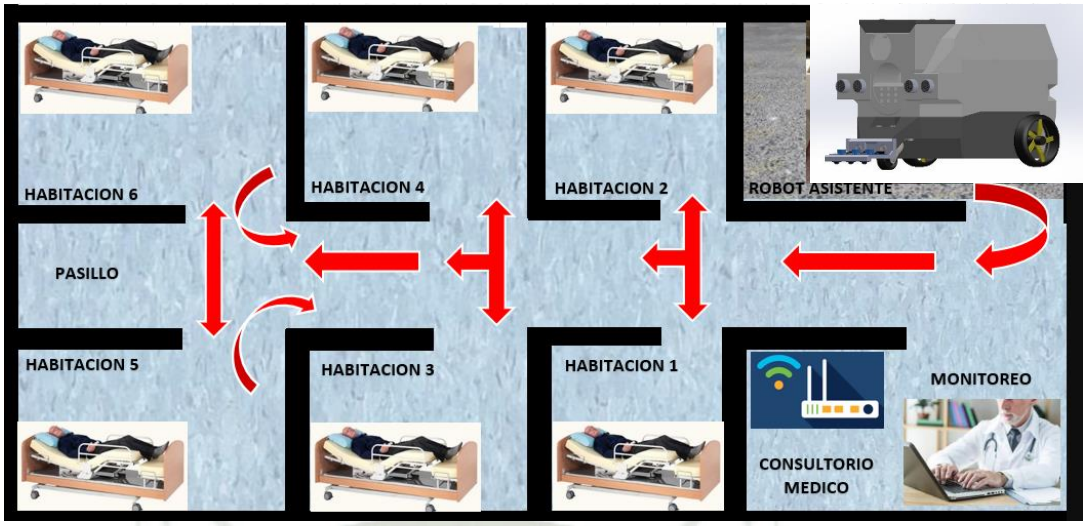


Figura 28. *Recorrido del robot móvil.*

Fuente: Elaboración propia.

3.1.4.1. Funciones a realizar el robot móvil

a. Además del robot, es necesario una Laptop y que ambos estén conectados a la misma red inalámbrica (WIFI) que proporciona el dispositivo Router del área.
b. Desde su Laptop, el personal de salud accede a la aplicación VNC para enlazarse con el robot móvil (la tarjeta Raspberry Pi 4).
c. Una vez enlazado, como vemos en la siguiente figura, en el menú del escritorio escogemos la opción PROGRAMACION – THONNY PYTHON IDE y se ejecuta el programa.

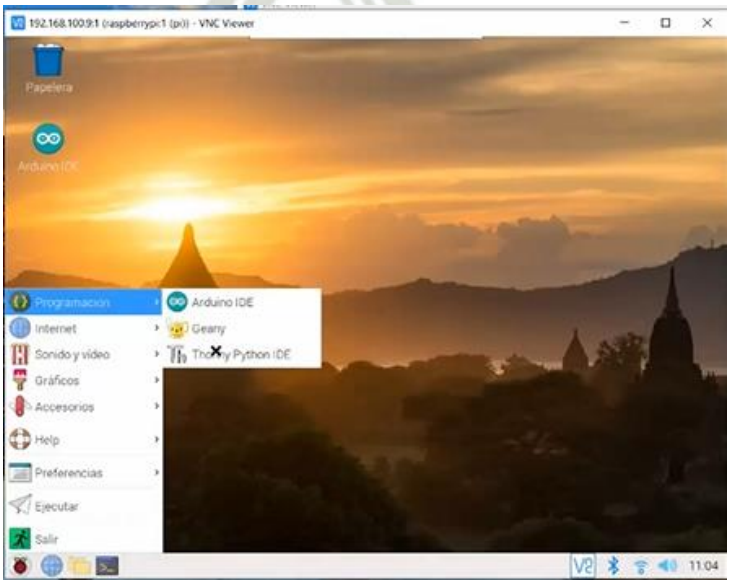


Figura 29. *Interfaz de Raspberry Pi 4 - Enlazamiento del robot móvil mediante la aplicación VNC.*

Fuente: Elaboración propia.

Nota. En la figura se muestra el enlazamiento con el robot móvil.

d. Se abrirá un panel (interfaz de usuario), con el cual, el médico o personal de salud controla las funciones del robot y su desplazamiento. Éste último se logra gracias a una guía representada por conos azules desde el punto de inicio de recorrido hasta su punto final.

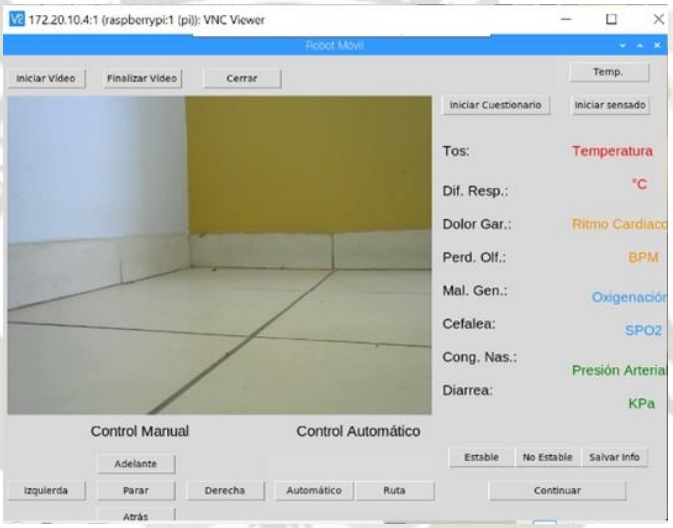


Figura 30. *Panel de control del robot*.

Fuente: Elaboración propia.

Nota. En la figura se muestra una captura del panel de control del prototipo de robot móvil.

e. Cuando el robot ingresa a la habitación, automáticamente se inicia el monitoreo del paciente de dos formas, si dicho paciente se encuentra consciente o inconsciente:

1) Si el paciente está consciente, se acerca el robot al paciente. Luego se inicia el cuestionario para hacerle preguntas típicas en cuanto a sus síntomas por COVID. Una vez que termina de responder, inicia el sensado para tomar los

signos vitales: toma de temperatura, la presión arterial, oxigenación, ritmo cardíaco y temperatura. Finalmente, el robot se retira y continua su camino.

2) Cuando el paciente está inconsciente, se acerca el robot al paciente. Luego se inicia el cuestionario para hacerle preguntas típicas en cuanto a sus síntomas por COVID. Si el robot, después de un tiempo de espera, no recibe respuesta, solo toma la temperatura y continúa su camino.

f. Todos los datos son visualizados por el personal de salud y almacenados en su computadora.

g. El robot continuará el recorrido y monitoreo planteado hasta que el personal de salud lo detenga.

3.1.4.2. ¿Cuándo trabaja en modo automático?

La forma de operar el robot móvil se puede configurar a dos modos: el primero permite la operación de manera remota a través de los controles “adelante, atrás, izquierda, derecha y parar” dentro de la interfaz de usuario, y se define como “manual”; el segundo se define como “automático”, el cual es activado mediante el botón “automático” dentro de la interfaz, y así, comienza a buscar al paciente para evaluarlo. Estos modos de operación son activados de manera manual, a través de los controles en la interfaz de usuario mencionados previamente.

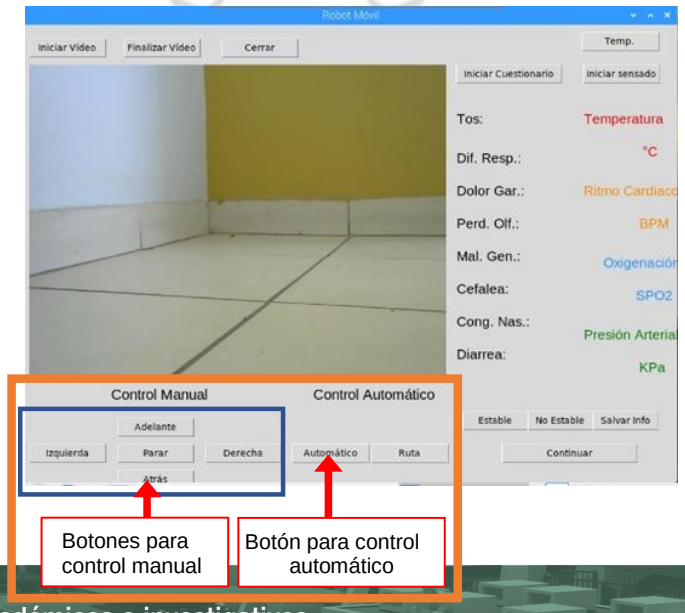


Figura 31. *Ventana de control manual o control automático.*

Fuente: Elaboración propia.

Nota. En la figura se puede observar una captura de pantalla de la ventana de control del robot móvil.

3.1.4.3. ¿En qué momento transmite la información, la almacena en el robot?

El robot primeramente obtiene la información de los diferentes sensores (pulsadores rojo, verde, sensor de temperatura, sensor de presión arterial, y sensor de oxigenación y ritmo cardíaco), donde los valores obtenidos de éstos son almacenados como variables dentro del código, es decir, por cada valor sensado se genera una variable; éstas permiten a su vez la visualización de los datos recabados de un paciente en la interfaz de usuario, y son mostrados en la misma una vez que se haya concluido el proceso de sensado. Un proceso de monitoreo es activado cuando se oprime alguno de los botones: “Iniciar cuestionario” e “iniciar sensado” dentro de la interfaz, que permiten la obtención de información con respecto a los síntomas y signos vitales del paciente respectivamente. A pesar de que los datos obtenidos de un paciente sean visualizados en la interfaz de usuario (como se muestra en la imagen siguiente), esto no indica que los datos se encuentren guardados en un archivo; para llevar a cabo esto último se debe presionar el botón “Salvar info”, el cual almacena en un archivo .csv las respuestas del cuestionario, así como los signos vitales obtenidos del paciente. Los archivos guardados se encuentran en la carpeta “archivo pacientes” ordenados de acuerdo con la fecha de escritura, esta carpeta se ha colocado en el escritorio para un fácil acceso.

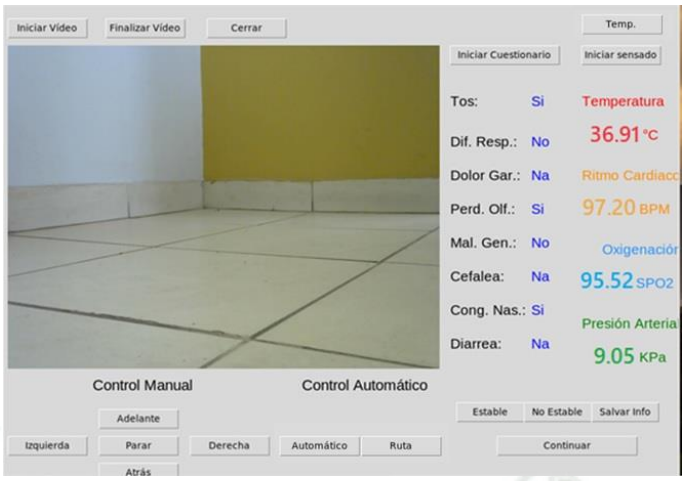


Figura 32. *Valores de sensores.*

Fuente: Elaboración propia.

En la Figura 33 se muestra la visualización de un archivo de paciente generado, en el cual se pueden observar las respuestas obtenidas por el paciente de acuerdo con las preguntas realizadas, así como los signos vitales: temperatura, ritmo cardíaco, oxigenación y presión arterial.

	A	B	C
1		Preguntas	Respuestas
2	0	Tos	Si
3	1	Dificultad para respirar	No
4	2	Dolor de garganta	Si
5	3	Perdida de olfato	Na
6	4	Malestar general	No
7	5	Cefalea	Na
8	6	Congestión nasal	Si
9	7	Diarrea	Na
10	8	Temperatura (°C)	36.91
11	9	Ritmo cardiaco (Bpm)	97.2
12	10	Oxigenación (Spo2)	95.52
13	11	Presión arterial (Kpa)	9.05
14	12	Fecha y hora	15:50:28, 15/11/21

Figura 33.*Archivo generado por el robot móvil.*

Fuente: Elaboración propia.

3.1.4.4. ¿El software en CPU guarda la información recibida?

La tarjeta Raspberry Pi 4 es una computadora, como cualquier computadora trabaja por medio de un sistema operativo. Al ejecutar el programa diseñado para el robot móvil, la información de los sensores primeramente se guarda en la memoria RAM, al momento de permanecer como variable dentro del programa; y posteriormente, al presionar el botón “salvar info” se guarda en la memoria del sistema (memoria SD) en forma de archivo csv.

Luego, para guardar la información de la memoria en la misma PC del personal de salud se puede compartir vía Internet usando Google Drive.

3.1.4.5. ¿Cómo funciona para llevar material médico de un ambiente a otro?

El robot consta de una cavidad donde se puede colocar material médico y otros insumos: mascarillas, alcohol, medicamentos, etc. Siguiendo la guía azul, no solo podría llevar todo esto en un solo ambiente, si no también, entre diferentes ambientes.

Para esto, se puede extender la guía azul (representada por conos azules u otros objetos acordes al espacio) hacia el objetivo deseado para llevar y traer material médico entre diferentes ambientes.

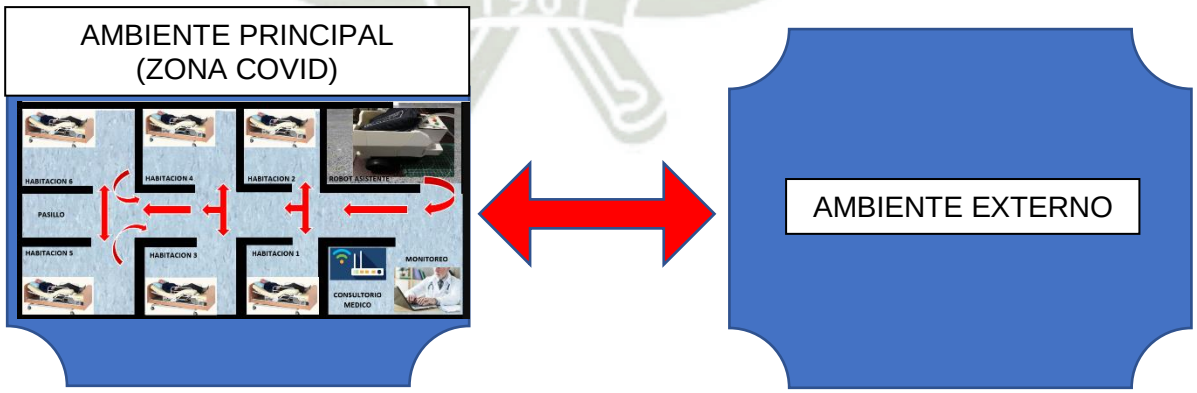


Figura 34. *Desplazamiento extra.*

Fuente: Elaboración propia.

3.2. Comparación y selección del lenguaje de programación (Python)

A continuación, se presenta una tabla comparativa entre diferentes lenguajes de programación (Tabla 4) detallando sus diferencias, ventajas y desventajas.

Tabla 4.

Ventajas y desventajas de los lenguajes de programación.

Lenguaje de programación	Paradigma	Ventajas	Desventajas
LABVIEW[1]	Imperativo	– Fácil de aprender – Reutilización de código	– Menos capacidad de procesamiento – Seguridad (bajo nivel de acceso)
C++[2]	Imperativo	– Multiplataforma – Alto Rendimiento – Veloz – Variedad de librerías	– Lenguaje muy amplio – Código ocupa bastante memoria
PYTHON[3]	Imperativo	– Software libre – Portable – Flexible	– Lentitud al ejecutar múltiples hilos – Problemas con hosting

Fuente. Elaboración propia.

Evaluando los lenguajes de programación mostrados en la Tabla 4, se decidió tener como lenguaje principal PYTHON. Debido a que es un software libre, portable, y flexible para procesar información de otros dispositivos. Además de un costo accesible.

[1] Paradigmas de Programación. Comparación de los lenguajes de DAtaflow LabVIEW y VEE. (Espinosa Muñoz, 2015)
[2] Qué es C++ : Características y aplicaciones (OpenWebinars, 2019)
[3] Python, el lenguaje de programación más popular de 2020. (Mochales Mielgo, 2020)

3.3. Comparación y selección del microcontrolador

A continuación, se presenta una tabla comparativa de diversos microcontroladores seleccionados, así detallaremos sus ventajas y desventajas de cada uno. Donde resaltaremos algunas características que son deseables en un microcontrolador capaz de controlar un prototipo robot como asistente para pacientes COVID.

Tabla 5.

Ventajas y desventajas del microcontrolador.

MICROCONTROLADOR	LENGUAJE	VENTAJAS	DESVENTAJAS
Arduino[4]	C++	– Bajo Costo – Prototipo Rápido de proyectos – Múltiples entradas y salidas – Código abierto	– Procesamiento limitado – Poca memoria de almacenamiento
Raspberry pi[5]	Python	– Simplificado y rápido – Portable – Cuenta con sistema operativo	– Sensibles a la estática – Menos puertos USB
PIC[6]	Ensamblador C++	– Tamaño pequeño – Variedad de modelos – Bajo costo	– Susceptible al ruido electromagnético – Software especializado

Fuente. Elaboración propia.

[4] Arduino ventajas y desventajas (Gonzales, 2021)
[5] Programación en Python – Nivel básico. (COVANTEC, 2019) , Conociendo Raspberry Pi (Pez Nuss, 2020)
[6] Microcontroladores, Universidad Fermín Toro. Facultad de Ingeniería. Escuela de Telecomunicaciones. (Virgúez Javier, 2016)

Los microcontroladores expuestos son ideales para nuestro proyecto, sin embargo, tomando en cuenta las diferentes funciones que se ejecutan y el complejo procesamiento de información, escogemos Raspberry pi 4. Gracias a su fuerte capacidad de procesamiento, es adecuado para proyectos más complejos y que requieren de un sistema operativo completo.

Además, para un mejor desempeño podemos añadir a Arduino (ATMega328). Microcontrolador que nos apoyará con el desplazamiento del robot móvil.

3.3.1. Ventajas de la Raspberry pi 4 modelo b con respecto a sus predecesores

El modelo B de cuatro núcleos de la Pi 4 B es más rápido y potente que su predecesor, el modelo B+ de Raspberry Pi 3. Para aquellos interesados en los puntos de referencia, la CPU de la Pi 4, el procesador principal de la placa ofrece de dos a tres veces el rendimiento del procesador de la Pi 3 en algunos puntos de referencia.

A diferencia de su predecesora, la nueva placa es capaz de reproducir vídeo de 4K a 60 fotogramas por segundo, lo que mejora las credenciales del centro multimedia de Raspberry.

Esto no quiere decir que todo el vídeo se reproducirá sin problemas. El soporte de esta aceleración de hardware para vídeo con codificación H.265 es actualmente un proyecto en curso para los distintos sistemas operativos de la marca, por lo que se trata de una característica potencial de futuro más que algo disponible en la actualidad.

La Raspberry Pi 4 modelo B también soporta internet inalámbrico desde el primer momento, con Wi-Fi y Bluetooth incorporados. Sin embargo, estos son sólo los sistemas operativos recomendados oficialmente, y una gran variedad de otros sistemas operativos también funcionan en la Pi.

Especificaciones de la Raspberry Pi 4 modelo B

- **Sistema en un chip:** Broadcom BCM2711
- **CPU:** Procesador de cuatro núcleos a 1,5 GHz con brazo Cortex-A72
- **GPU:** VideoCore VI
- **Memoria:** 1/2/4GB LPDDR4 RAM

- **Conectividad:** 802.11ac Wi-Fi / Bluetooth 5.0, Gigabit Ethernet
- **Vídeo y sonido:** 2 x puertos micro-HDMI que admiten pantallas de 4K@60Hz a través de HDMI 2.0, puerto de pantalla MIPI DSI, puerto de cámara MIPI CSI, salida estéreo de 4 polos y puerto de vídeo compuesto.
- **Puertos:** 2 x USB 3.0, 2 x USB 2.0
- **Alimentación:** 5V/3A vía USB-C, 5V vía cabezal GPIO
- **Expansión:** Cabezal GPIO de 40 pines
- Doble banda de wifi que nos permite conectar a 2.4 GHz y 5GHz
- Compatible con lenguaje Python (éste no se necesita comprar una licencia, ya que es un software libre)

3.4. Sensores

3.4.1. Comparación y selección del sensor de presión arterial.

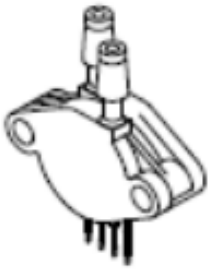
	MPX2050d	MPX10DP
Sensor		
Voltaje de operación	*10V DC*	*3V DC*
Corriente de trabajo	6mA	6mA
Rango de presión	0 a 50 kPa	0 a 10 kPa
Sensibilidad	0.8 mV/kPa	3.5 mV/kPa
Presión	0.1	0.1

Tabla 6. Comparación entre MPX2050 y *MPX10DP*

Fuente: NXP (2021) y NXP (2008)

3.4.1.1. Justificación de la selección de sensor de presión MPX10DP

Usamos este sensor dentro de nuestro sistema de control neumático para tomar la presión arterial. Los MPX10DP son sensores de presión piezo resistivos de silicio que proporcionan una salida de voltaje muy precisa y lineal, directamente proporcional a la presión aplicada.

- Opera a un voltaje de 3 VDC – 6 VDC.
- Tiene una corriente de trabajo de 6mA.
- Rango de presión: 0 a 10 kPa
- Sensibilidad de 3.5 mV/kPa
- Precisión de presión de 0.1
- Tiempo de respuesta 1 ms

Estos sensores son de buena precisión y su voltaje de operación se adapta a la alimentación proveída por la tarjeta Raspberry Pi 4 (3.3 V). Además, son de bajo costo y alta disponibilidad en el mercado.

3.4.2. Comparación y selección del sensor de oxigenación y frecuencia cardíaca

	MAX30100	MAX30102	MAX30105
Sensor			
Voltaje de operación	3.1VDC a 5*V DC*	3.1VDC a 5*V DC*	3.1VDC a 5*V DC*
Corriente de trabajo	0.6mA	0.6mA	0.6mA
Funcionalidad	Pulsioxímetro y medición del ritmo cardíaco, sector salud	Pulsioxímetro de alta sensibilidad	Detector de humo de alta sensibilidad

Potencia de radiación del Led infrarrojo	6.5mW	6.5mW	6.5mW
Potencia de radiación del Led Rojo	9.8mW	9.8mW	9.8mW
Sensibilidad	640 a 980 nm en un área de 1.36 mm^2		640 a 980 nm en un área de 1.36 mm^2
Dimensiones	25.4mm x 12.7mm	14mm x 17mm	50mm x 30mm
Temperatura de operación	-40°C a +85°C	-40°C a +85°C	-40°C a +85°C

Tabla 7. Comparación entre MAX30100, MAX30102 y MAX30105

Fuente: Sparkun (s.f.). Naylamp Mechatronics (s.f.). (Maxim integrated, 2020)

3.4.2.1. Justificación de la selección de sensor para ritmo cardiaco y saturación de oxígeno MAX 30100

Módulo sensor MAX30100, permite medir la frecuencia cardiaca y el porcentaje de saturación de oxígeno de la hemoglobina (SaO2) en la sangre utilizando un circuito fotoeléctrico (pulsioximetría no invasiva).

El dispositivo integra combina dos LED, un fotodetector, óptica optimizada y procesamiento de señales analógicas de bajo ruido para detectar señales de pulsioximetría y frecuencia cardíaca.

- Resolución ADC: 14 bits
- Longitud de onda del Led Infrarrojo 870 nm - 900 nm
- Tensión directa: 1.4 V
- Longitud de onda del Led Rojo 650 nm - 670 nm
- Tensión directa: 2.1 V
- Sensibilidad del Fotodetector de 640 a 980 nm en un área de 1.36 mm^2

El fotodetector mide la cantidad de luz reflejada a través del dedo de la persona, siendo que la sangre oxigenada absorbe más luz infrarroja, y la poco oxigenada más luz roja, a lo cual el sensor detecta la luz para calcular la concentración de oxígeno.

La operación de energía ultra baja aumenta la duración de la batería:

- Tasa de muestreo programable y corriente LED para ahorro de energía
- Corriente de apagado ultra baja (0.7 µA)

La funcionalidad avanzada mejora el rendimiento de la medición:

- Cancelación de luz ambiental integrada.
- Capacidad de frecuencia de muestreo alta.
- Capacidad de salida de datos rápida.

El MAX30100 es totalmente configurable a través de registros de software, y los datos de salida digital se almacenan en un FIFO de 16 profundidades dentro del dispositivo. El FIFO permite que el MAX30100 se conecte a un microcontrolador o microprocesador en un bus compartido, donde los datos no se leen continuamente desde los registros del dispositivo.

El SpO2 ADC es un convertidor sigma delta de sobre muestreo de tiempo continuo con una resolución de hasta 16 bits. La velocidad de datos de salida del ADC se puede programar de 50 Hz a 1 kHz. El MAX30100 incluye un filtro de tiempo discreto patentado para rechazar la interferencia de 50 Hz/60 Hz y el ruido ambiental residual de baja frecuencia.

El MAX30100 integra controladores LED rojos e IR para impulsar pulsos LED para mediciones de SpO2 y HR. La corriente del LED se puede programar de 0 mA a 50 mA (típico solamente) con el voltaje de suministro adecuado. El ancho de pulso del LED se puede programar de 200 µs a 1,6 ms para optimizar la precisión de la medición y el consumo de energía según los casos de uso.

Este sensor es recomendado para el monitoreo médico (aprobado por DIGEMID Dirección General de Medicamentos del Perú).

3.4.3. Comparación y selección del sensor de temperatura

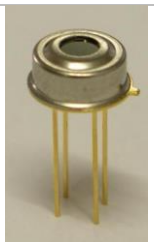
	DTH11	AMG8833	MLX90614
Sensor			
Voltaje de operación	3.3V *DC*.-5.5V *DC*	3.3V *DC*	*5V DC*
Corriente de trabajo	1.5mA	4.5mA	1.3mA
Precisión	2 °C	2.5 °C	0.2 °C
Rango de temperatura	0°C a 50°C	0°C a 80°C	-70°C a 380°C
Dimensiones	12mm x 5.5m x 23.5mm	8mm x 11.6mm x 4.3mm	9mm x 9mm x 17mm

Tabla 8. *Comparación entre DTH11, AMG8833* y MLX90614

Fuente: (ADSONG, 2020) (PANASONIC, 2020) (MELEXIS, 2020)

3.4.3.1. Justificación de la selección de sensor de temperatura AMG8833 (cámara térmica)

Usamos este sensor de matriz 8x8 de sensores infrarrojos de alta precisión, de tipo termopila, para adquirir la temperatura.

- Detección de temperatura de área bidimensional: 8 × 8 (64 píxeles).
- Salida digital.
- Opera a un voltaje de 3 VDC – 3.6 VDC.
- Rango de temperatura de objeto de medición: 0 °C a 80 °C
- Sensibilidad: 0.05 °C – 0.16 °C
- Distancia de detección humana: 7m o menos

- Ángulo de visión: 60°
- Interfaz externa: I2C.
- Cuadros por segundo: 10
- Resolución de salida de temperatura: 0.25 °C

AMG8833 detecta la radiación de rayos IR emitida por los pacientes. Cuando este sensor se conecta al Raspberry Pi 4, devolverá una matriz de 64 lecturas individuales de temperatura infrarroja sobre I2C, de las cuales, se elegirá la más alta.

Actúa como una sofisticada cámara térmica, pero lo suficientemente compacta y simple para una fácil integración.

Estos sensores son de buena precisión, capaces de detectar a una persona hasta 7 metros de distancia. Su voltaje de operación se adapta a la alimentación proveída por la tarjeta Raspberry Pi 4 (3.3 V)

3.4.4. Comparación y selección del sensor ultrasónico

	LV MAX SONAR	SRF05	HC-SR04
Sensor			
Voltaje de operación	*2.5V DC a 5.5V DC*	5V *DC*	5V *DC*
Corriente de trabajo	2mA a 3mA	30mA a 50mA	15mA
Comunicación	TTL	TTL	TTL
Rango de lectura	0 – 645 cm	3 – 300 cm	2 – 400 cm
Dimensiones	19.9mm x 22.1mm x 15.5mm	43mm x 20mm x 17mm	45mm x 20mm x 15mm

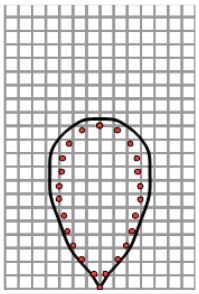
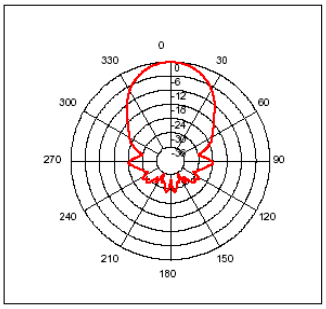
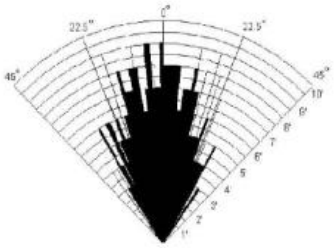
Ángulo de medición			

Tabla 9. *Comparación entre LV MAX SONAR, SRF05 y* HC-SR04

Fuente: en (MaxBotix, 2020), (ALLDATASHEET, 2020), (ELEC FREAKS, 2020), ROBOT - HEAD TO TOE (2013).

3.4.3.1. Justificación de la selección de sensor ultrasónico HCSR 04

Usamos este sensor ultrasónico con el fin de detectar obstáculos y evitarlos. El módulo sensor HCSR-04, pequeño sensor ultrasónico de buena precisión, bajo costo y bajo consumo energético, permite medir distancia mediante ultrasonido, para determinar la distancia a la que se encuentra un objeto de 2 a 400 cm.

- Frecuencia: 40 Hz
- Ángulo de medición: 15°
- Rango de lectura: 2 – 400 cm
- Precisión: 0.3 cm

Posee dos transductores, un emisor y un receptor piezoeléctricos. Donde el emisor emite 8 impulsos de ultrasonido (40KHz) luego de recibir la orden en el pin TRIG, el sonido luego de viajar rebota y es detectado por el receptor piezoeléctrico, luego el pin ECO cambia a Alto (5V) por un tiempo igual al que demora en la onda desde ser emitida hasta ser detectada en el rebote, el tiempo del pulso ECO se mide con el microcontrolador y calcula la distancia del objeto. No le afecta la luz, ni el color negro, aunque con ciertos materiales blandos acústicamente puede ser difícil que los detecte.

Su voltaje de operación se adapta a la alimentación proveída por microcontrolador ATMega328 (5V)

3.4. Actuadores

3.5.1. Motorreductor motor PO4826

3.5.1.1. Pololu metal gear motor 25d 6-12vdc

La placa frontal tiene dos orificios de montaje roscados para tornillos M3. Utilizaremos el soporte de motorreductor metálico de 25 mm diseñado a medida (que se muestra en la imagen a continuación) para montar el motorreductor en su proyecto a través de estos orificios de montaje y los tornillos que vienen con el soporte.

Figura 35. *Motorreductor Pololu 25D mm.*

Fuente: ECKSTEIN (2020).

Figura 36. *Soporte de motorreductor metálico Pololu 25D mm. par de soportes.*

Fuente: ECKSTEIN (2020).

El eje de salida de la caja de engranajes de 4 mm de diámetro funciona con el cubo de montaje de aluminio universal Pololu para ejes de 4 mm, que se puede usar para montar nuestras ruedas Pololu más grandes (60 mm, 70 mm, 80 mm y 90 mm de diámetro) o

ruedas y mecanismos personalizados para el eje de salida del motorreductor como se muestra a continuación:

Figura 37. *Motorreductor metálico de 25D mm. adaptador de rueda.*

Fuente: ECKSTEIN (2020).

Figura 38. *Adaptador de rueda hexagonal de 12 mm para eje de 4 mm en un motorreductor de metal de 20 D mm.*

Fuente: ECKSTEIN (2020).

3.5.2. Controlador de motor L298

- Tensión de alimentación de funcionamiento hasta 46V
- Corriente CC total hasta 4 A.
- Baja tensión de saturación
- Protección de sobre temperatura
- Tensión de entrada lógica “0” hasta 1,5V (Inmunidad al ruido alto)

3.5.2.1. Descripción

El L298 es un circuito monolítico integrado en 15 paquetes de plomo Multiwatt y PowerSO20. Es un controlador de doble puente completo, de alto voltaje y corriente, diseñado para aceptar niveles lógicos TTL para controlar cargas inductivas como relés, solenoides, motores paso a paso y CC. Se proporcionan dos entradas para habilitar o deshabilitar el dispositivo, las entradas son independientes una de otra. Los emisores de los transistores más bajos de cada puente están conectados entre sí y la terminal externa correspondiente puede utilizarse para la conexión de un resistor / sensor externo.

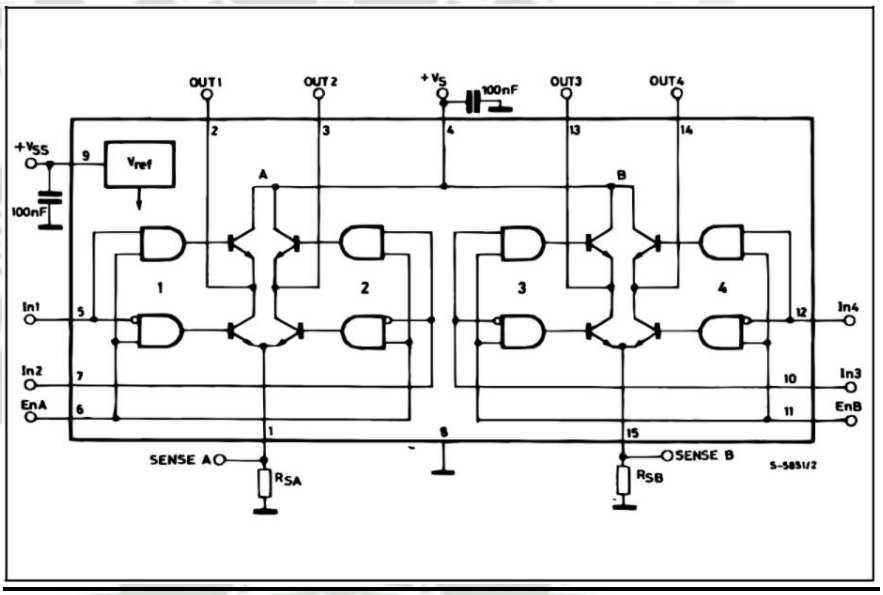


Figura 39. *Diagrama de bloques.*

Fuente: ALLDATASHEET (2020).

Símbolo	Parámetro	Valor	Unidad
V_s	Fuente de alimentación	50	v
V_{ss}	Voltaje de suministro lógico	7	V

V_I, V_{en}	Voltaje de entrada y habilitación	-0,3 a 7	V
I_o	Corriente de salida máxima (cada canal) • No repetitivo (t=100α s) • Repetitivo (80% en -20% de descuento; t_{en}=10 ms) • Operación CC	 3 2.5 2	 A A A
V_{sens}	Voltaje detección	-1 a 2,3	V
P_{tot}	Disipación total de energía (caso T=75°C)	25	w
T_{op}	Temperatura de funcionamiento de la unión	-25 a 130	°C
T_{stg}, T_j	Temperatura de almacenamiento y unión	-40 a 150	°C

Tabla 10. *Parámetros*
Fuente. (ALLDATASHEET, 2020)

3.5.3. Cámara web HD C505

El modelo C505 es una cámara Web con vídeo HD 720p y un micrófono de gran alcance que permite mantener una conversación con claridad y sin alzar la voz hasta una distancia de 3 metros. Además, su cable USB-A extralargo de 2 metros ofrece opciones de montaje versátiles y únicas.

3.5.3.1. Requisitos del sistema

- Compatible con Windows 7 o posterior, macOS 10.7 o posterior, Chrome OS
- Un puerto USB-A

3.5.3.2. Especificaciones técnicas

- Resolución máxima: 720p/30 fps

- Cámara megapíxel: 1.2
- Tipo de enfoque: Fijo
- Tipo de lente: Plástico
- Micrófono integrado: mono
- Radio de micrófono: Hasta 2.74 m
- Campo visual diagonal (dFoV): 60°
- Clip de montaje universal listo para laptops, LCD o monitores

Figura 40. *Cámara web HD C505.*

Fuente: ALLDATASHEET (2020).

3.5.4. Batería de LiPo

Las baterías LiPo (abreviatura de Litio y polímero) son baterías recargables, compuesta en ocasiones de múltiples celdas usadas en aplicaciones que requieren corrientes superiores a 1A con bajo peso y tamaño reducido, por ejemplo, sistemas de radio control, como aviones, helicópteros, drones, cámaras, celulares, linternas, entre otros.

- **Vida útil:** 2 a 3 años o unas 500 cargas completas
- **Formas:** son fabricadas en diversas formas y tamaños
- **Eficiencia:** mejor relación tamaño eficiencia que otras tecnologías

- **Tasa de descarga:** Alta tasa de descarga desde 1A hasta más de 25A
- **Voltaje de Celda:** Cada celda tiene 3.7V, y se puede encontrar baterías de 1 a 6 celdas

La capacidad indica cuanta corriente puede suministrar la batería y se mide en miliamperios por Hora (mAh). Es una manera de indicar la cantidad de carga medida en miliamperios que puede suministrar la batería durante 1 hora antes que se descargue completamente. Por ejemplo, una batería LiPo de 1000 mAh sería completamente descargada en una hora con una demanda de 1000 miliamperios. Si ésta misma batería tenía una demanda de 500 miliamperios tomaría 2 horas para descargarla.

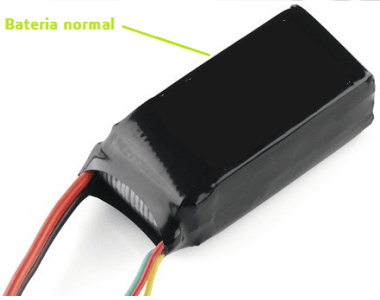


Figura 41. *Batería de LiPo.*

Fuente: Elaboración propia.

3.5.5. Conversor analógico digital MCP3008

El chip MCP3008 es un conversor de 8 canales con 10 bits de resolución. Está formado por 16 pines, 8 para la recogida de datos analógicos, la alimentación se conecta a través de los pines VDD y DGND, el pin AGND se utiliza para circuitos de alta precisión y se conectaría a la masa analógica. VREF es utilizado para modificar la escala de las entradas analógicas. También encontramos los pines de CLK (reloj), CS (selector de chip), DOUT (salida digital) y DIN (entrada digital).

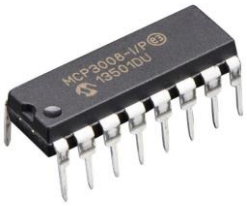


Figura 42. *Conversor analógico-digital MCP3008.*

Fuente: Sánchez Lanau (2018).

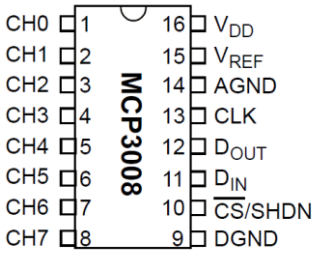


Figura 43. *Pines del ADC MCP3008.*

Fuente: Sánchez Lanau (2018).

3.5.6. Integrado ULN2003

El ULN2003 es un arreglo monolítico de transistores Darlington de alto voltaje y corriente. Consiste en siete pares Darlington NPN que cuentan con salidas de alto voltaje con diodo de abrazadera de cátodo común para conmutar cargas inductivas. La clasificación de corriente de colector de un solo par Darlington es 500mA. Los pares Darlington pueden combinarse para una mayor capacidad de corriente. Las aplicaciones incluyen controladores de relé, controladores de martillo, controladores de lámpara, controladores de pantalla (descarga de gas LED), controladores de línea y búfer lógicos. el ULN2003 tiene una resistencia base de la serie 2.7kΩ para cada par Darlington para operar directamente con dispositivos TTL o CMOS de 5V.

3.5.6.1. Características

- No. de pares Darlington NPN: 7
- Corriente de colector nominal de 500 mA (salida única)
- Salidas de alto voltaje hasta de 50 V
- Las entradas son compatibles con varios tipos lógica.
- Aplicación: controlador de relé.
- Encapsulado: DIP 16 pines.

Figura 44. *ULN 2003.*

Fuente: Elaboración propia.

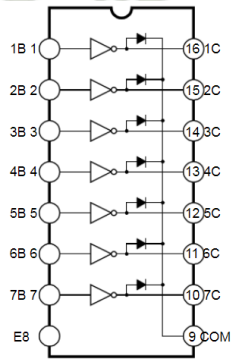


Figura 45. *Pines del ULN2003*

Fuente: Elaboración propia.

3.5.7. Capacidad para llevar 1Kg de peso

Para el desplazamiento del presente proyecto, los motores deben ser capaces de cargar consigo mismo y con 1Kg adicional. En la siguiente tabla vemos que el prototipo tiene un peso total de 2478.9 g, lo que equivale a 2.5 Kg.

COMPONENTES	PESO (g)
Rasberry Pi 4	50
Tarjeta ATMega328	50
Cámara Logitech	75
Cámara Raspberry 5 MP	3
Electroválvula	18
Batería LiPo	147
Powerbank	150
Parlante Bluetooth	200
Sensor de saturación de oxigenación y ritmo cardíaco	50
Sensor de temperatura	2.9
Sensor de presión	5
Sensor ultrasónico	10 x 4 = 40
ULN2003	1
MCP3008	1
L298D	1
Motores	95 x 3 = 285
Estructura	1450
TOTAL	**2478.9**

Tabla 11. *Peso de componentes electrónicos*
Fuente: Elaboración propia

Esto significa que, en total, ambos motores deben tener la capacidad de llevar 2.5 Kg + 1 kg = 3.5 Kg. Ahora, analizando la hoja de datos de un motor Pololu PO4826, tenemos lo siguiente:

Voltaje	Velocidad (RPM)	Corriente (mA)	Extrapolación de parada		Potencia máxima (W)
			Torque (kg.mm)	Corriente (A)	
6 V	80	100	75	2	1.5

Tabla 12. *Datos de motor Pololu PO4826*
Fuente: Elaboración propia

Para saber cuánto peso puede llevar un motor, usamos la fórmula de Torque para determinar la fuerza con la que funciona:

$$\tau = F.r$$

$$F = \frac{\tau}{r} = \frac{75\ kg.mm}{65 \div 2\ mm} = 2.3\ kg$$

Si un motor puede llevar 2.3 kg, ambos llevarían un total de 4.6 kg. Lo que sería suficiente para 3.5 kg que se había previsto.

CAPÍTULO IV

DISEÑO, CONSTRUCCIÓN Y PUESTA EN MARCHA DEL PROTOTIPO

4.1. Especificaciones técnicas

ESPECIFICACIONES TÉCNICAS			
MECÁNICA	Tracción	Diferencial	
	Peso	200 gramos	
	Dimensiones	26 x 18 x 15 cm	
ELECTRÓNICA	Microprocesador	Quad Core ARM A72	Raspberry Pi 4
	Microcontrolador	ATMega328	Arduino
	Sensores (*)	Presión	MPX10DP
		Temperatura	AMG8833
		Ultrasónico	HCSR04
		Ritmo cardiaco y saturación de oxígeno.	MAX30100
		Pulsadores	1 x 1 cm
	Actuadores	Motores DC y válvula solenoide.	6 - 7 V
		Cámara web (para detectar conos azules)	HD C505
		Cámara Raspberry Pi 3/ Pi 4 (para detección de rostros)	- 5 MP - 25 mm x 20 mm x 9 mm
		Parlante	Bluetooth
	Alimentación	Power Bank (circuito)	5 V, 3 Ah
		Batería LiPo (motores)	7.4 V, 2.8 Ah
	Consumo	Voltaje	5 V
		Corriente	1.4 A
		Potencia	7 W
	Tiempo de autonomía	14 horas	Tiempo de autonomía

(*) NOTA: Detalle de los sensores usados en la siguiente sección.

Tabla 13. *Ficha técnica*
Fuente: Elaboración propia

4.2. Consumo del circuito

A continuación, identificamos los voltajes de operación de cada uno de los componentes electrónicos que vamos a utilizar y también el consumo de corriente.

	Voltaje de operación (v)	Consumo de corriente (mA)
Raspberry Pi 4	5	1000
Sensor temperatura AMG8833	3.3	4.5
Sensor de presión arterial MPX10DP	3.3	6
Sensor de oxigenación y frecuencia cardíaca MAX30100	3.3	0.6 – 1.2
ULN2003	3	0.9 -1.3
Cámara web HD C505	5	300
Cámara Raspberry Pi3/Pi4	5	80
ATMega328	5	2.7
Sensores ultrasónicos HC-SR04	5	15
L298D	5	36
Motores	6-12	2000

Tabla 14. *Consumo de corriente y voltaje*
Fuente: Elaboración propia

4.3. Diseño y cálculos para el desplazamiento

El mecanismo de desplazamiento para el proyecto será de tipo uniciclo, una estructura que consta de dos ruedas fijas convencionales sobre el mismo eje, controladas de manera independiente, y una rueda loca que le confiere estabilidad (Bambino, 2008).

Las ruedas fijas se anexan a 2 motores Pololu PO4826, cuyo diagrama podemos ver a continuación:

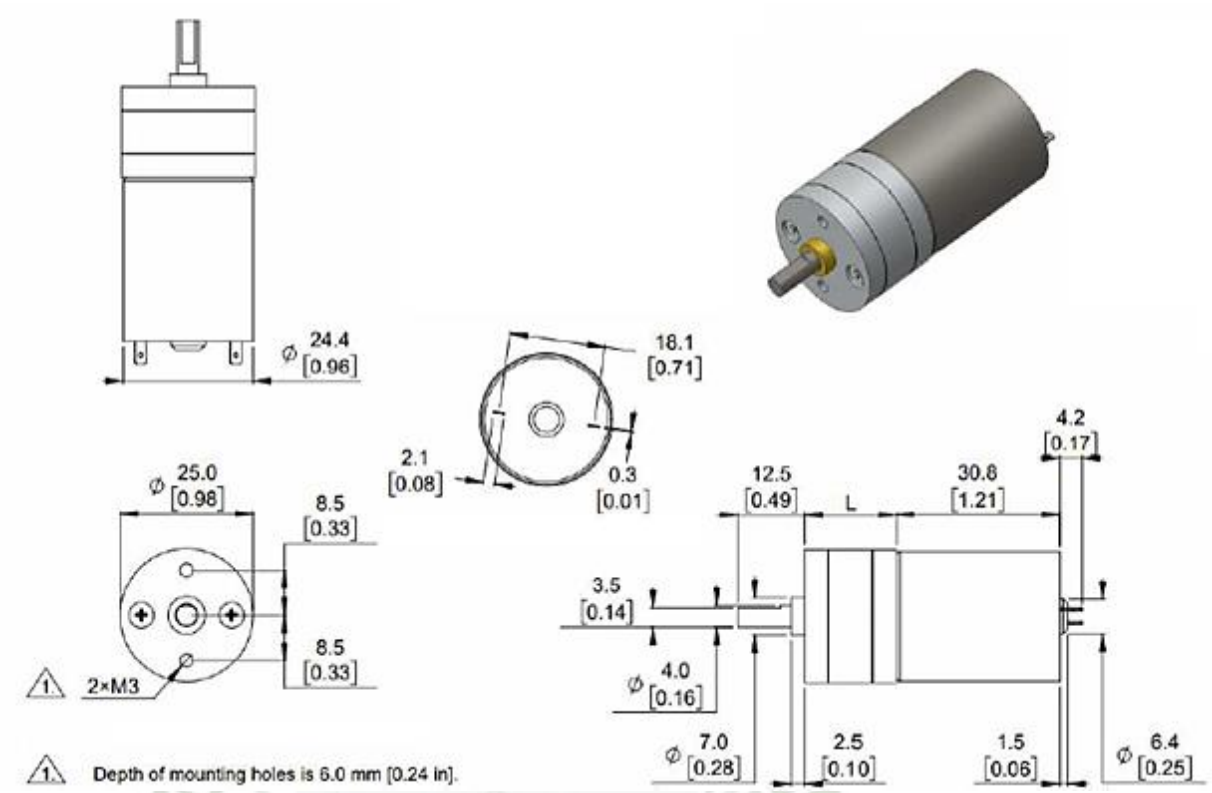


Figura 46. Diagrama del motor

Fuente: DataSheet PO4826.

4.3.1. Cálculo de velocidad del motor

Para esto se consideran los siguientes datos:

- RPM del motor: 80 a 6V 2Amp.
- Diámetro de llanta: 65mm.
- Voltaje de alimentación para el motor: 7.4V 2.8Amp

Primero se debe obtener la longitud de circunferencia de la llanta:

$$L = \pi * \text{diametro} \rightarrow \pi * 65mm = 204mm = 20cm$$

Con lo anterior sabemos que una vuelta de la llanta equivale a recorrer 20cm aproximadamente.

Considerando la información del motor podemos ahora calcular cuánto recorrería en un minuto:

$$d = L * RPM = 20cm * 80 = 1600cm = 16Mtrs.$$

Idealmente el robot recorrería 16 metros en un minuto, ahora bien, el voltaje de alimentación empleado no son los 6V, por lo tanto, debemos ajustar los valores de voltaje.

Si en nuestro caso usamos una fuente de 7.4V, ¿a cuántos RPM equivale?, hacemos una ecuación por regla de tres simple:

$$6V - 80$$
$$7.4V - RPM$$

$$RPM = \frac{80 \times 7.4}{6} = 99$$

Y la distancia recorrida sería:

$$d = L * RPM = 20cm * 99 = 1980cm = 20\,Mtrs.$$

Es decir, el motor tiene una velocidad aproximada de 20 metros en un minuto.

4.3.2. Cálculos durante la marcha del motor y arranque del motor.

A continuación, a partir de los datos de voltaje V y corriente I_N del DataSheet que corresponde al motor Pololu PO4826, podemos encontrar la potencia P_N e impedancia R_N del motor durante el arranque. Luego, encontramos la corriente I_a , potencia P_a e impedancia R_a del motor durante el arranque.

Durante la marcha:

$$I_N = 2A$$

$$P_N = I_N \times V = (2A)(6\,V)$$

$$P_N = 12W$$

$$R_N = \frac{P_N}{(I_N)^2} = \frac{12W}{(2)^2} = 3\omega$$

Durante el arranque:

$$I_a = I_N \times 1.5$$

$$I_a = 3A$$

$$P_a = I_a \times V = (3A)(6\,V)$$

$$P_a = 18W$$

$$R_a = \frac{P_a}{(I_N)^2} = \frac{18W}{(3A)^2} = 2\omega$$

4.4. Diseño del Circuito Electrónico

A continuación, se presenta el diseño del circuito electrónico, de manera general y de acuerdo con cada parte que lo compone.

En la siguiente figura se presenta el circuito electrónico diseñado para la construcción del robot móvil. El cual cuenta con dos componentes principales, los cuales son: Raspberry Pi 4 y ATMega328. Estos se encargan de ejecutar las funciones del robot, teniendo como controlador principal a la tarjeta Raspberry Pi 4.

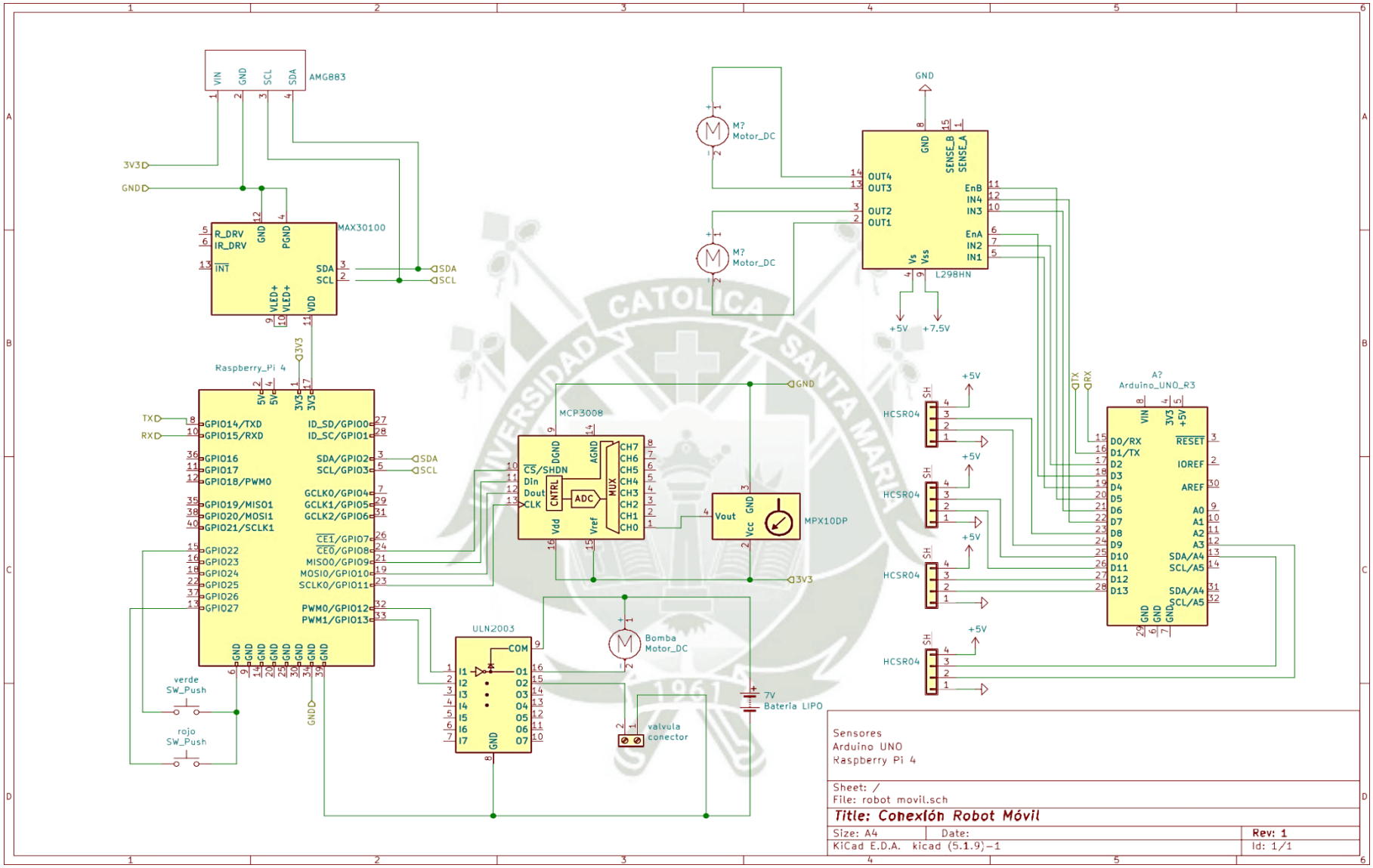


Figura 47. *Diagrama de conexiones del sistema electrónico.*
Fuente: Elaboración propia.

4.1. Circuito para la tarjeta Raspberry Pi 4

El circuito empleado para conectar los sensores y actuadores a la tarjeta Raspberry Pi 4 se presenta en la siguiente figura.

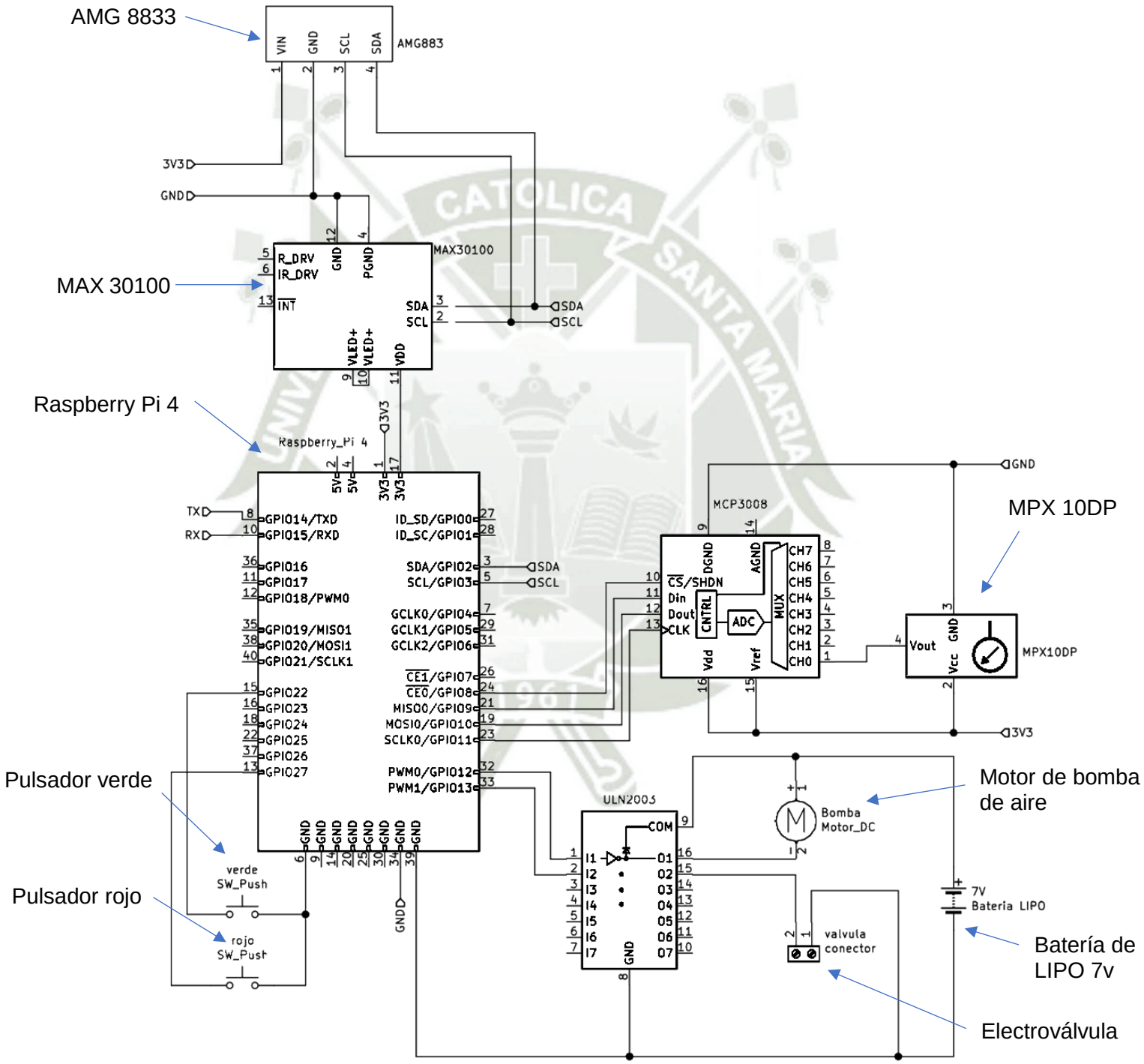


Figura 48. *Circuito para la tarjeta Raspberry Pi 4.*

Fuente: Elaboración propia.

Los sensores AMG8833, MAX30100, se conectan a la tarjeta Raspberry Pi 4 mediante el protocolo de comunicación i2c, así mismo estos sensores son alimentados a 3.3 Volts, a través del voltaje que otorga la misma tarjeta.

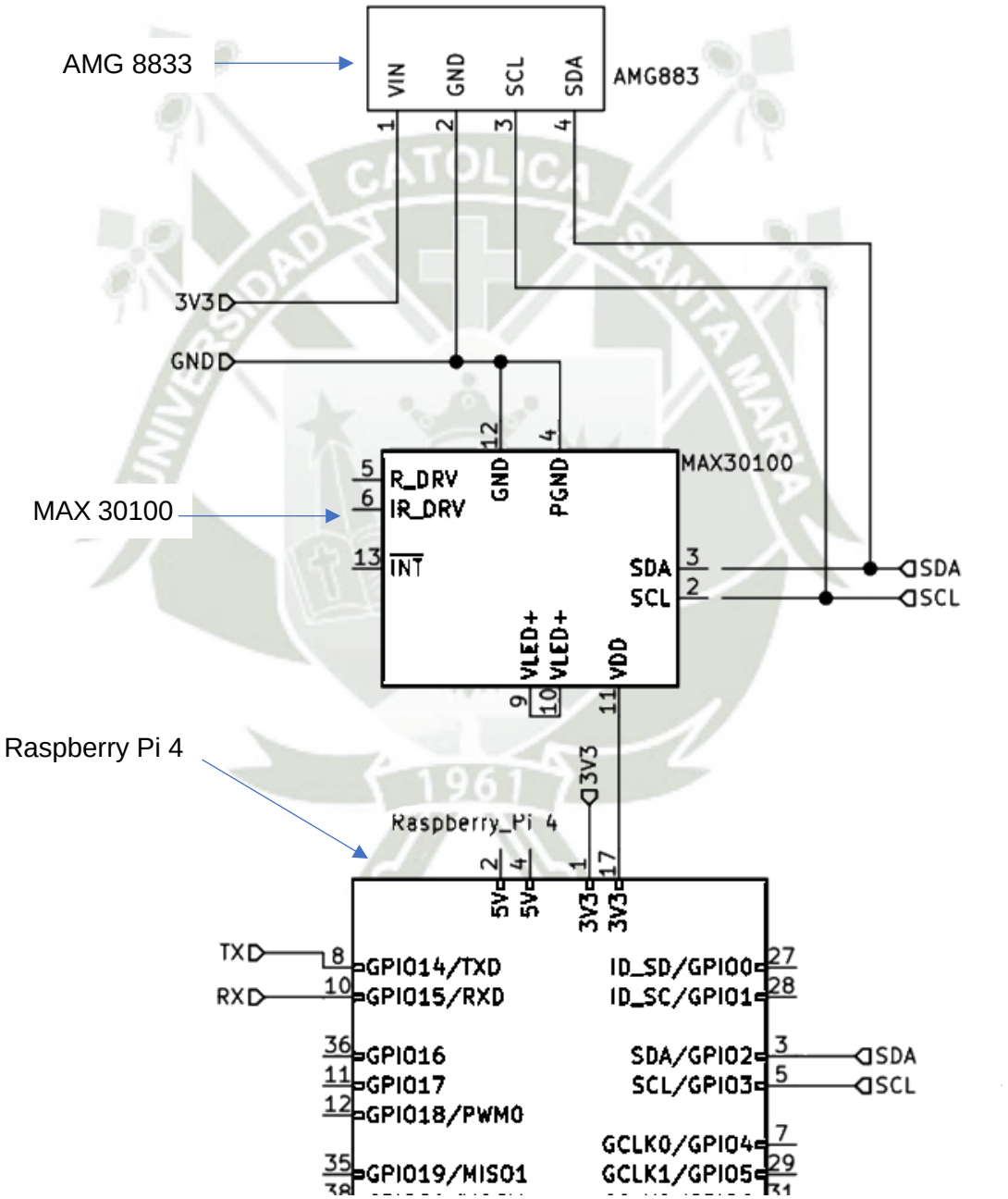


Figura 49. Conexión de sensores AMG8833 Y MAX30100.

Fuente: Elaboración propia.

Por otro lado, el sensor de presión de aire MPX10DP es de tipo analógico, debido a que la tarjeta no tiene un puerto con ADC, se emplea el integrado MCP3008, el cual es un convertidor analógico a digital, contiene ocho canales de entrada y la salida es enviada por el protocolo de comunicación ICSP. Es decir, para poder recibir la señal de presión de aire el integrado MCP3008 recibe la salida analógica del sensor MPX10DP, y posteriormente la envía hacia la tarjeta Raspberry Pi 4 para su lectura, ambos dispositivos son alimentados a 3.3 Volts proporcionados por la tarjeta Raspberry Pi 4.

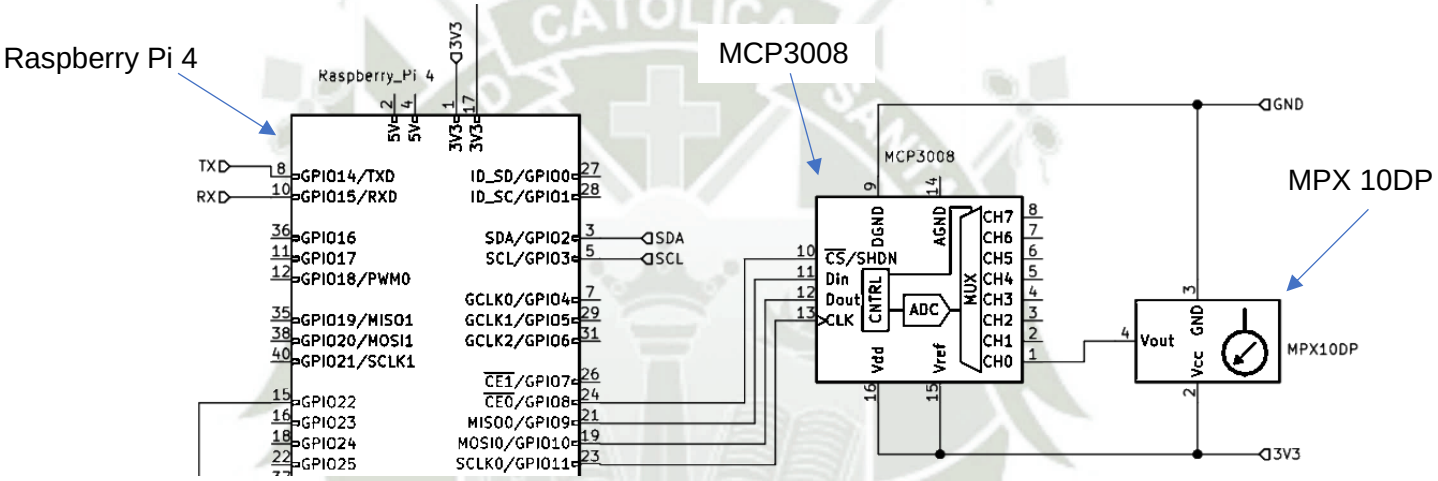


Figura 50. *Conexión del integrado MCP3008 y el sensor MPX10DP.*

Fuente: Elaboración propia.

Los pulsadores representados por los colores verde y rojo son empleados para poder recibir las respuestas del paciente que se esté atendiendo; los pulsadores son conectados a los puertos GPIO17, GPIO22 y GPIO27 de la Raspberry Pi 4, como se presenta en el diagrama siguiente, los cuales al cerrar el circuito permiten recibir una señal en bajo por el puerto al cual está conectado el pulsador.

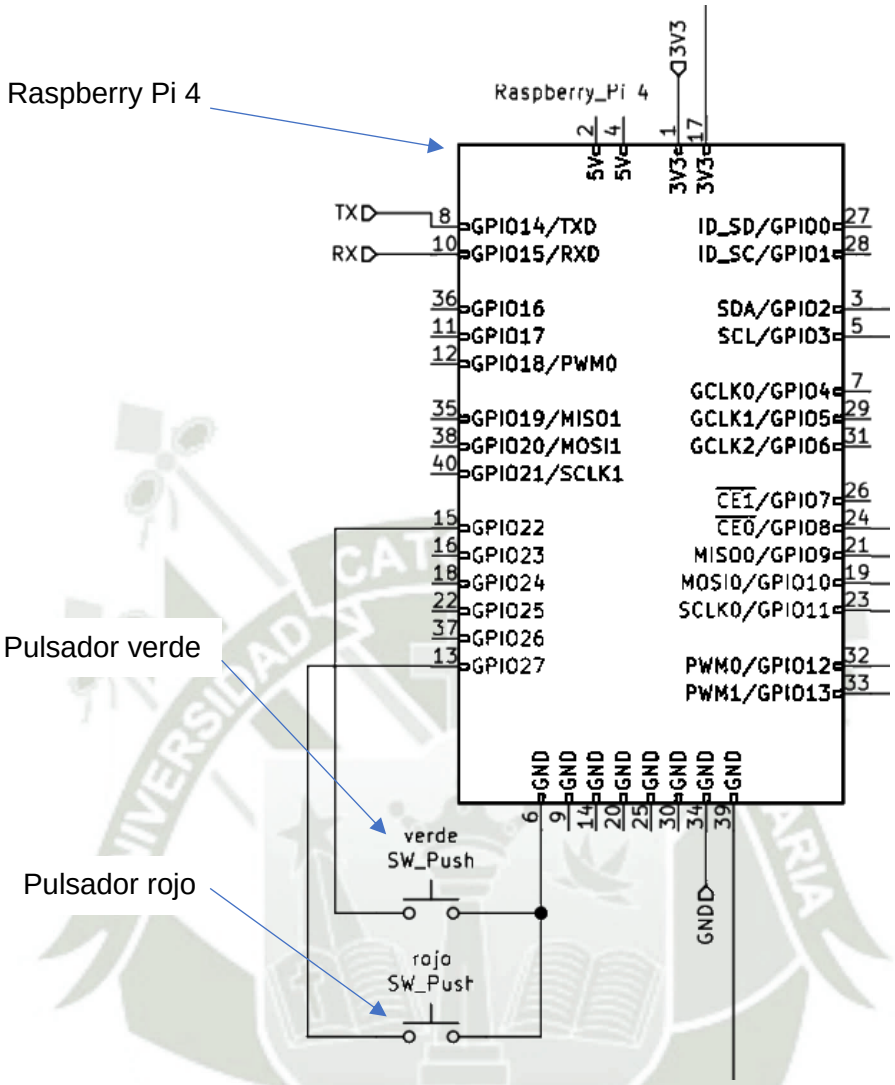


Figura 51. *Conexión de pulsadores.*

Fuente: Elaboración propia.

Lo anterior mencionado corresponde a la recepción de las señales de los diferentes sensores. Además de esto, también se envían señales de control para activar el motor de la bomba y electroválvula empleados para la etapa de medición de presión arterial. Para poder controlar los dispositivos de potencia mencionados, se hace uso del integrado ULN2003, el cual recibe dos señales provenientes de los puertos GPIO32 y GPIO33 de la tarjeta Raspberry Pi 4; la salida 1 del integrado ULN2003 se conecta a un conector del motor de la bomba de aire, y la salida 2 es conectada a un conector de la electroválvula.

Los dispositivos de potencia son alimentados por una batería de 7 volts. En la Figura 52 se presenta esta conexión.

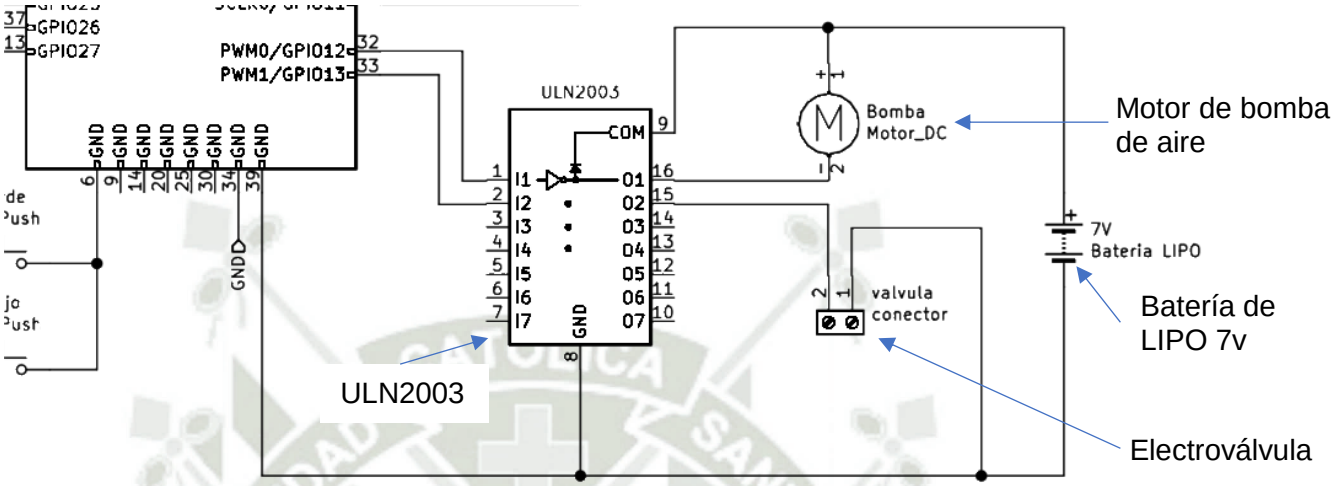


Figura 52. *Conexión de Dispositivos de potencia para presión arterial.*

Fuente: Elaboración propia.

4.2. Circuito para ATMega328

En el diagrama siguiente se presenta la conexión de diferentes dispositivos y la tarjeta ATMega328. La cual se encarga de controlar los motores de las llantas del robot, y también se encarga de detectar objetos cerca del robot mediante sensores de distancia (HCSR04). A continuación, se describe la conexión de los diferentes elementos.

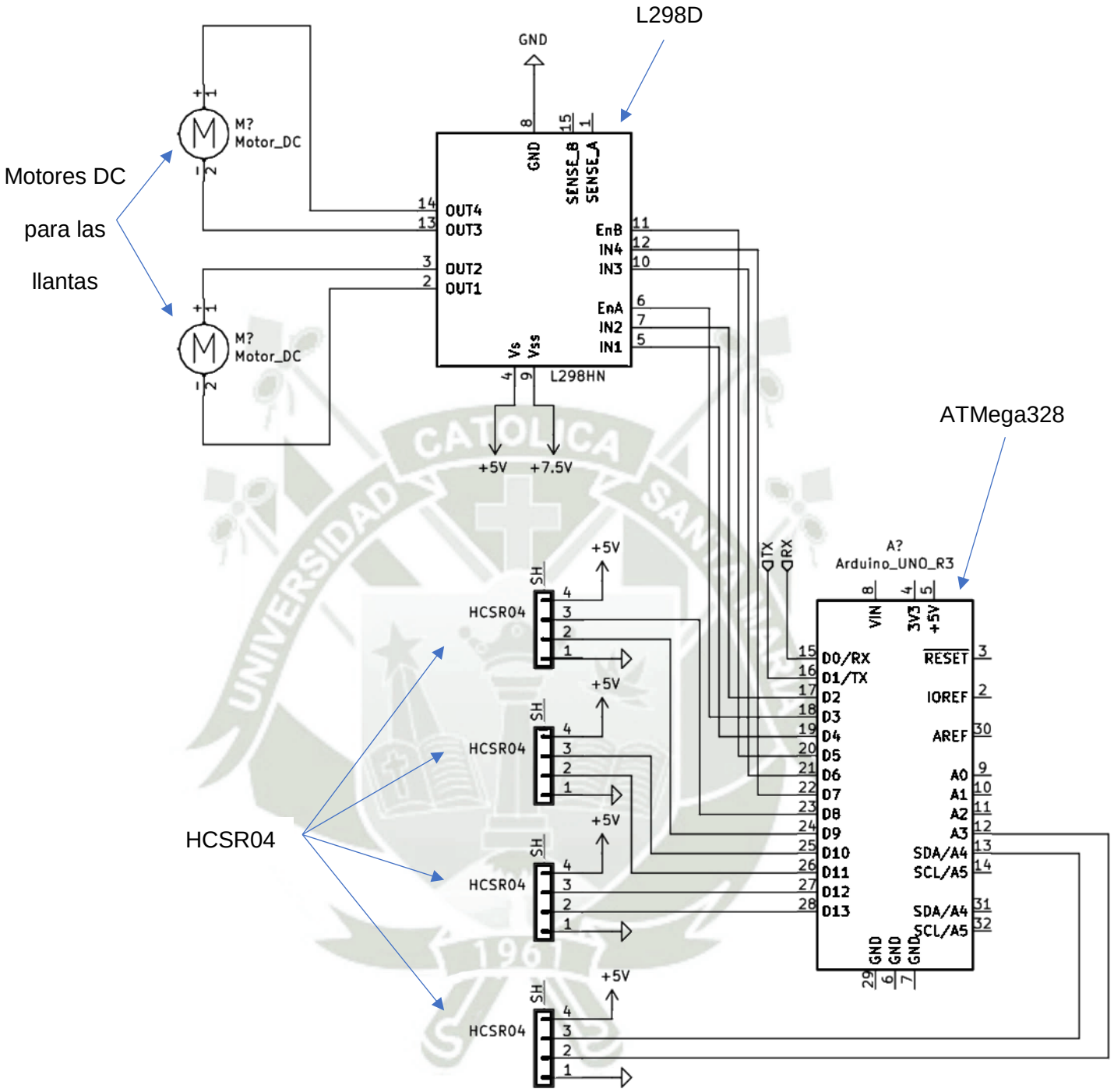


Figura 53. *Circuito para ATMega328.*

Fuente: Elaboración propia.

Para controlar los motores DC se hace uso del integrado L298D, donde los motores son alimentados por una batería de 7 Volts.

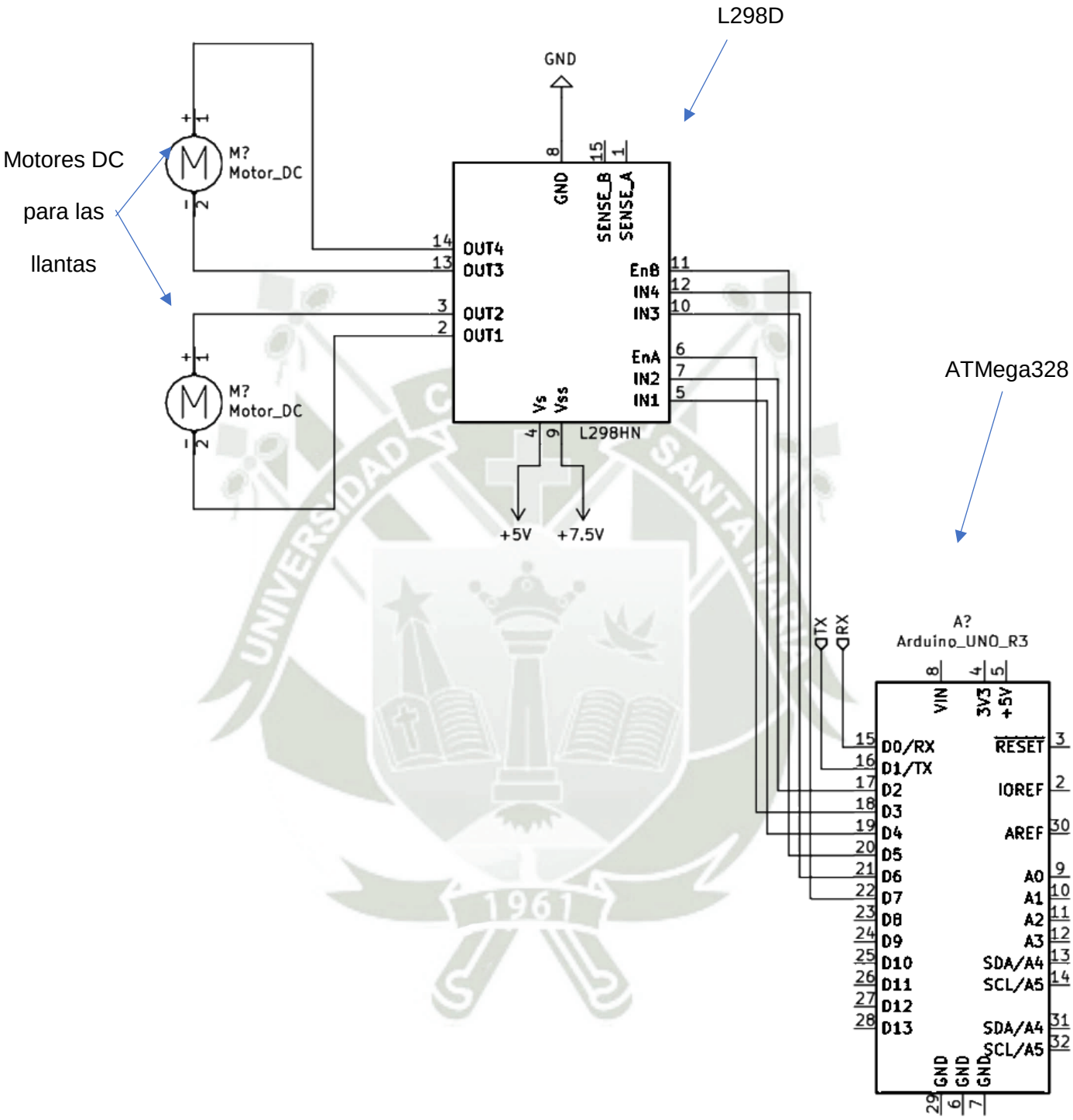


Figura 54. *Conexión de Motores DC.*

Fuente: Elaboración propia.

Para detectar obstáculos se emplean cuatro sensores HCSR04, los cuales son conectados de acuerdo con la Figura 55, y son alimentados con 5 Volts, mismos que son otorgados por la tarjeta ATMega328.

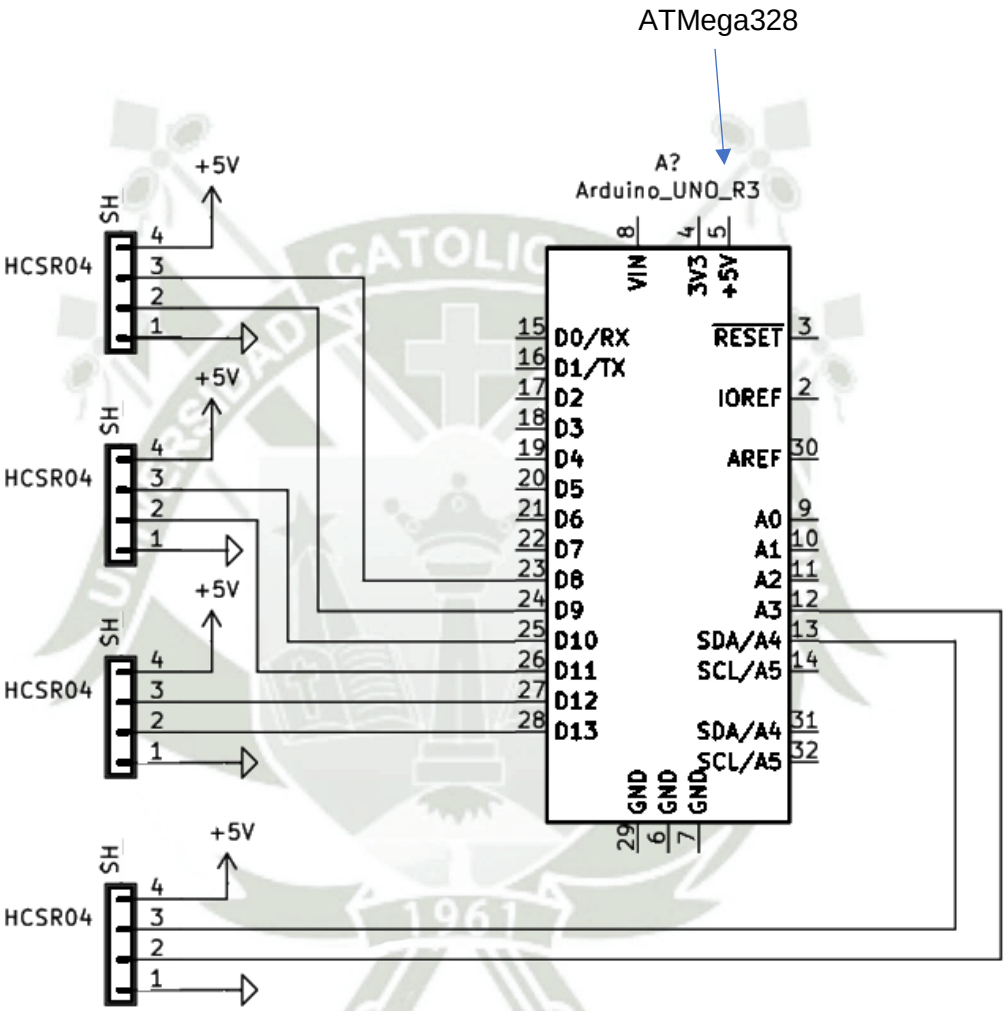


Figura 55.*Conexión de Sensores HCSR04.*

Fuente: Elaboración propia.

Se continúa ahora con la conexión eléctrica de los sensores y actuadores, así como, de los circuitos integrados a emplear para este proyecto.

En la Figura 57, se presenta la conexión de los sensores (parte izquierda) y actuadores (parte derecha), con la tarjeta Raspberry Pi 4.

En la Figura 58, se presenta la conexión de los sensores (parte izquierda) y actuadores (parte derecha), con la tarjeta ATMega328.

Ambas tarjetas presentan conexión serial UART por la que se comunican las tarjetas Raspberry Pi 4 y ATMega328, como se puede ver en la siguiente figura:

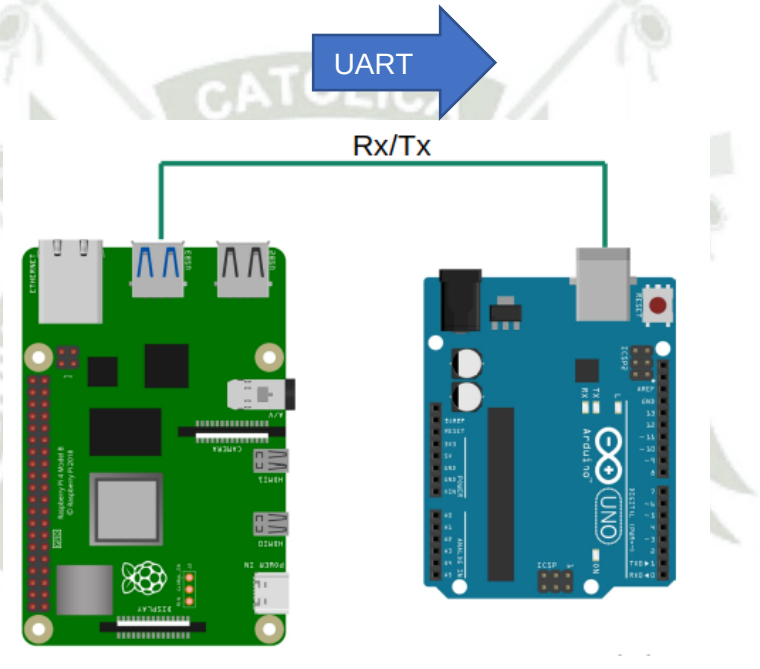


Figura 56. *Representación de comunicación serial, entre las tarjetas Raspberry Pi 4 y ATMega328.*
Fuente: Elaboración Propia.

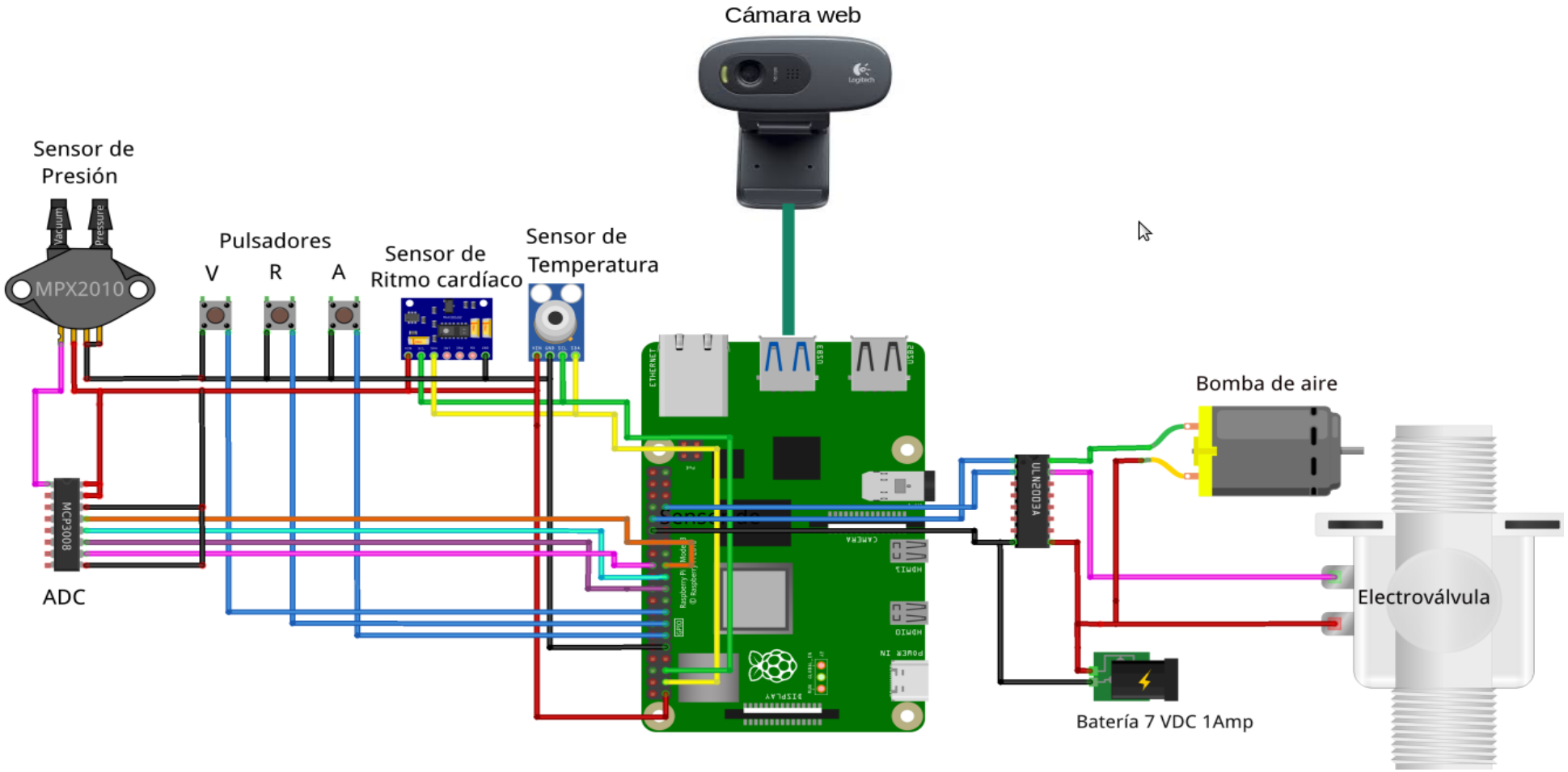


Figura 57. *Pre - ensamble del prototipo (conexión de sensores y actuadores en la tarjeta Raspberry Pi 4).*

Fuente: Elaboración Propia.

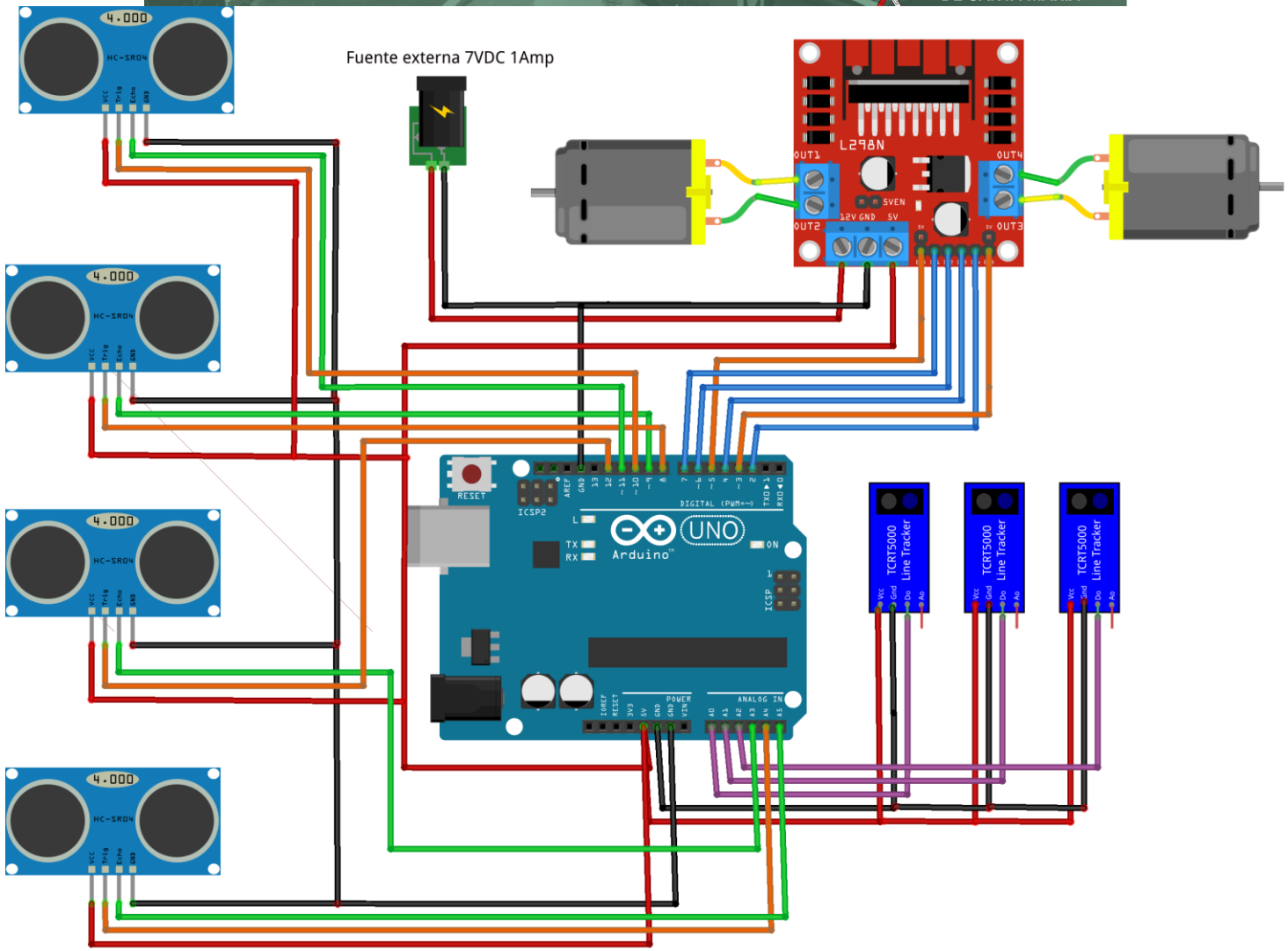


Figura 58. *Pre - ensamble del prototipo (conexión de sensores y actuadores en la tarjeta* ATMega328***)***

Fuente: Elaboración Propia.

4.3. Alimentación del circuito

El circuito electrónico está alimentado por dos baterías, una para la tarjeta Raspberry Pi 4 y otra para los actuadores de potencia (motores). La batería que alimenta a la tarjeta Raspberry Pi 4 es un Powerbank o banco de baterías, la cual otorga 5 Volts a 3 Amperes. Por otro lado, los dispositivos de potencia son alimentados por una batería de 7 Volts a 1 Ampere.

Los dispositivos: AMG8833, MAX30100, MCP3008, MPX10DP, UL2003 y pulsadores, son alimentados por el voltaje que otorga la tarjeta Raspberry Pi 4; mientras que los dispositivos: L298D y HCSR04, son alimentados por el voltaje que otorga la tarjeta ATMega328, y esta tarjeta es alimentada a su vez por la Raspberry Pi 4 mediante conexión por el puerto USB.

En el diagrama siguiente se presenta la distribución de la alimentación para los diferentes dispositivos, de la cual se puede concluir que el Powerbank alimenta a las tarjetas Raspberry Pi 4, ATMega328 y a los diferentes sensores e integrados empleados, mientras que la batería de LiPo alimenta a los motores, bomba de aire y electroválvula.

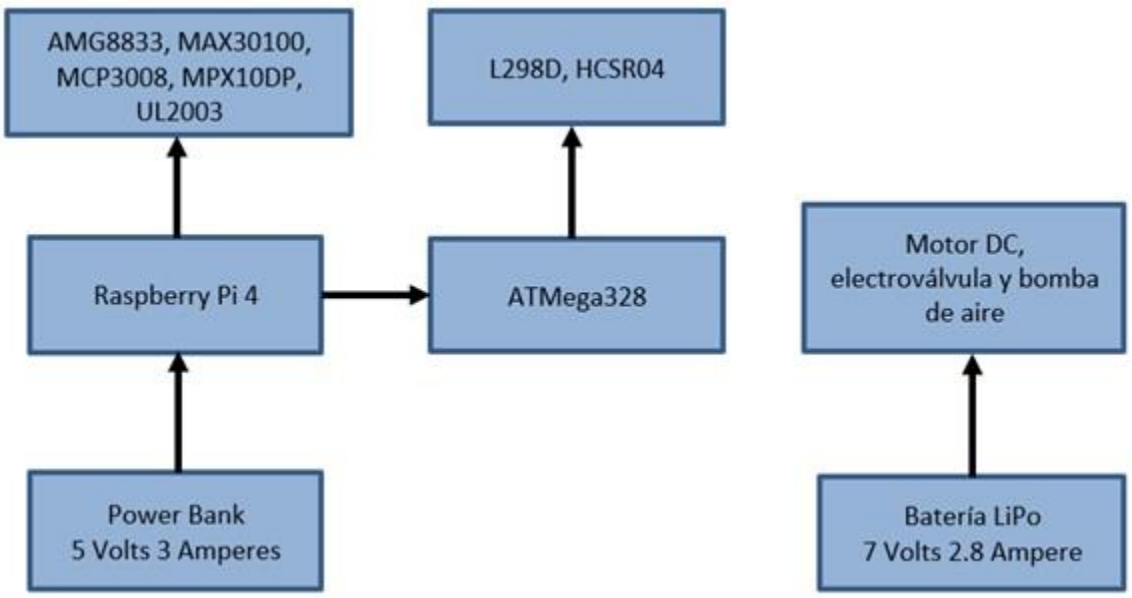


Figura 59. *Alimentación del circuito.*

Fuente: Elaboración Propia.

4.4. Consumo eléctrico

A continuación, se presenta la relación del consumo eléctrico del circuito.

- Consumo de corriente encendido de la tarjeta Raspberry Pi 4: 1400 mA
- Voltaje de operación de la tarjeta Raspberry Pi 4: 5V
- Consumo de corriente encendido de los motores: 2000 mA
- Voltaje de operación de motores: 6V

4.4.1. Cálculo de potencia y tiempo de autonomía con duración de las baterías

❖ **Batería PowerBank que alimenta el circuito principal.**

Para calcular el tiempo de duración de la batería PowerBank partimos del consumo eléctrico de la tarjeta Raspberry Pi 4, ya que ésta provee de corriente eléctrica a todo el circuito de sensores y tarjeta ATMega328, para lo cual es necesario calcular la potencia tanto de la tarjeta como de la batería usando la siguiente relación:

Potencia eléctrica (Watts) = Voltaje * Amperes

Por lo que el tiempo de duración de batería se obtiene por la siguiente ecuación:

$$T1 = \frac{Potencia\ eléctrica\ de\ la\ batería}{Potencia\ eléctrica\ de\ rpi\ 4}$$

La relación anterior también se puede expresar como:

$$T1 = \frac{Potencia\ de\ entrega\ de\ la\ batería}{Potencia\ de\ consumo}$$

Donde T1 es el tiempo estimado duración de la batería en horas.
La batería PowerBank entrega 5 V con una corriente máxima de 20 A.
Y el consumo del prototipo es de 5 V con una corriente de 1.4 A.
Por tanto, hacemos la conversión:

$$T1 = \frac{(5V)(20A)}{(5V)(1.4A)} = 14\ horas$$

El tiempo de autonomía sería de 14 horas.

❖ **Batería de LiPo que alimenta los motores DC.**

Para calcular el tiempo de duración de la batería LiPo partimos del consumo eléctrico de los motores de las ruedas y la bomba de aire, para lo cual es necesario calcular la potencia tanto de los motores como de la batería usando la siguiente relación:

Potencia eléctrica (Watts) = Voltaje * Amperes

Por lo que el tiempo de duración de batería se obtiene por la siguiente ecuación:

$$T2 = \frac{Potencia\ de\ entrega\ de\ la\ batería}{Potencia\ de\ consumo\ de\ los\ motores}$$

Donde T2 es el tiempo estimado duración de la batería en horas.

La batería LiPo entrega 7.4 V con una corriente máxima de 2.8 A.

Y el consumo del prototipo es de 6 V con una corriente de 2 A.

Por tanto, hacemos la conversión:

$$T2 = \frac{(7.4V)(2.8A)}{(6V)(2A)} = 2\ horas$$

La batería de LiPo usada en el proyecto se considera solo para demostración.

Por tanto, para que este tiempo T2 logre acercarse a T1 y así lograr un desempeño equivalente de 14 horas, se puede colocar hasta 7 baterías en paralelo.

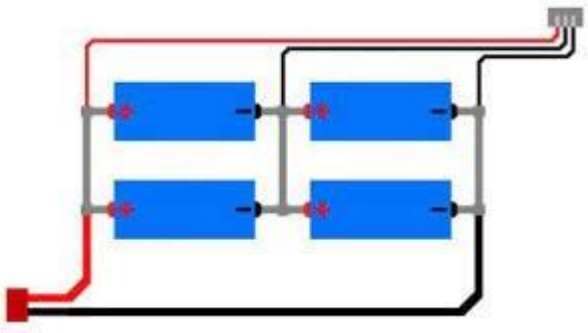


Figura 60. *Baterías LiPo en paralelo*

Fuente: Elaboración Propia.

4.5. Rediseño de Hardware

Con el fin de que el robot llegue a su destino automáticamente:

Para el desplazamiento automático, hemos adecuado la función de la cámara web HD C505 como un dispositivo de reconocimiento facial.

Además, se ha integrado en nuestro proyecto una cámara Raspberry Pi 3/ Pi 4 de 5 MP, otorgándole la función de reconocimiento y seguimiento de las camas de los pacientes (con una guía representada por conos azules).

4.5.1. Cámara Raspberry Pi 3/ Pi 4 de 5 MP

El módulo cámara de 5 megapíxeles está diseñado específicamente para Raspberry Pi, con un lente de foco fijo. Es capaz de tomar imágenes estáticas de 2592 x 1944, y también es compatible con el formato de video 1080p30, 720p60 y 640x480p60/90. Se conecta al Raspberry Pi 4 por medio de un pequeño conector en la parte superior de la tarjeta y utiliza la interfaz dedicada CSi, diseñado especialmente para la conexión de cámaras.

La placa en sí es muy pequeña, alrededor de 25 mm x 20 mm x 9 mm. Se conecta a Raspberry Pi 4 por medio de un cable de cinta corta.

La cámara es compatible con la última versión del Raspbian, el sistema operativo preferido de Raspberry Pi.

Figura 61. *Cámara Raspberry Pi 4*

Fuente: Datasheet.

CARACTERÍSTICAS

- 1,4 µm X 1,4 µm píxeles con tecnología OmniBSI de alto rendimiento (alta sensibilidad, baja diafonía, ruido bajo)
- tamaño óptico de 1/4 “
- funciones de control de imagen automáticas:
    - Control automático de exposición (AEC)
    - Balance de blancos automático (AWB)
    - Filtro de banda automático (ABF)
    - Calibración del nivel de negro automático (ABLC)
- controles programables para la velocidad de fotogramas, AEC / AGC 16 zonas / posición / control de peso, espejo y lado, recorte, ventanas, y el panorama
- puerto de vídeo digital (DVP) Interfaz de salida en paralelo
- 32 bytes de memoria programable una sola vez incorporada (OTP)

ESPECIFICACIONES TÉCNICAS

- Resolución: 1/4 5M
- Apertura: 2.9
- Longitud focal: 3.29
- FOV: 65 grados
- Tipo de sensor: OmniVision OV5647 Color CMOS QSXGA (5 megapíxeles)
- Tamaño del sensor: 3.67 x 2.74 mm (formato 1/4″)
- Cantidad de píxeles: 2592 x 1944
- Tamaño de píxel: 1.4 x 1.4 um
- Lente: f = 3.6 mm, f / 2.9
- Ángulo de visión: 54 x 41 grados
- Campo de visión: 2.0 x 1.33 m a 2 m
- Lente SLR de fotograma completo equivalente: 35 mm
- Enfoque fijo: 1 m hasta el infinito
- Video: 1080p a 30 fps con códec H.264 (AVC)
- Video de hasta 90 fps en VGA
- Tamaño de la placa: 25 x 24 mm (sin incluir el cable flexible)

❖ **Reensamblado con las cámaras HD C505 y RASPBERRY 5 MP**

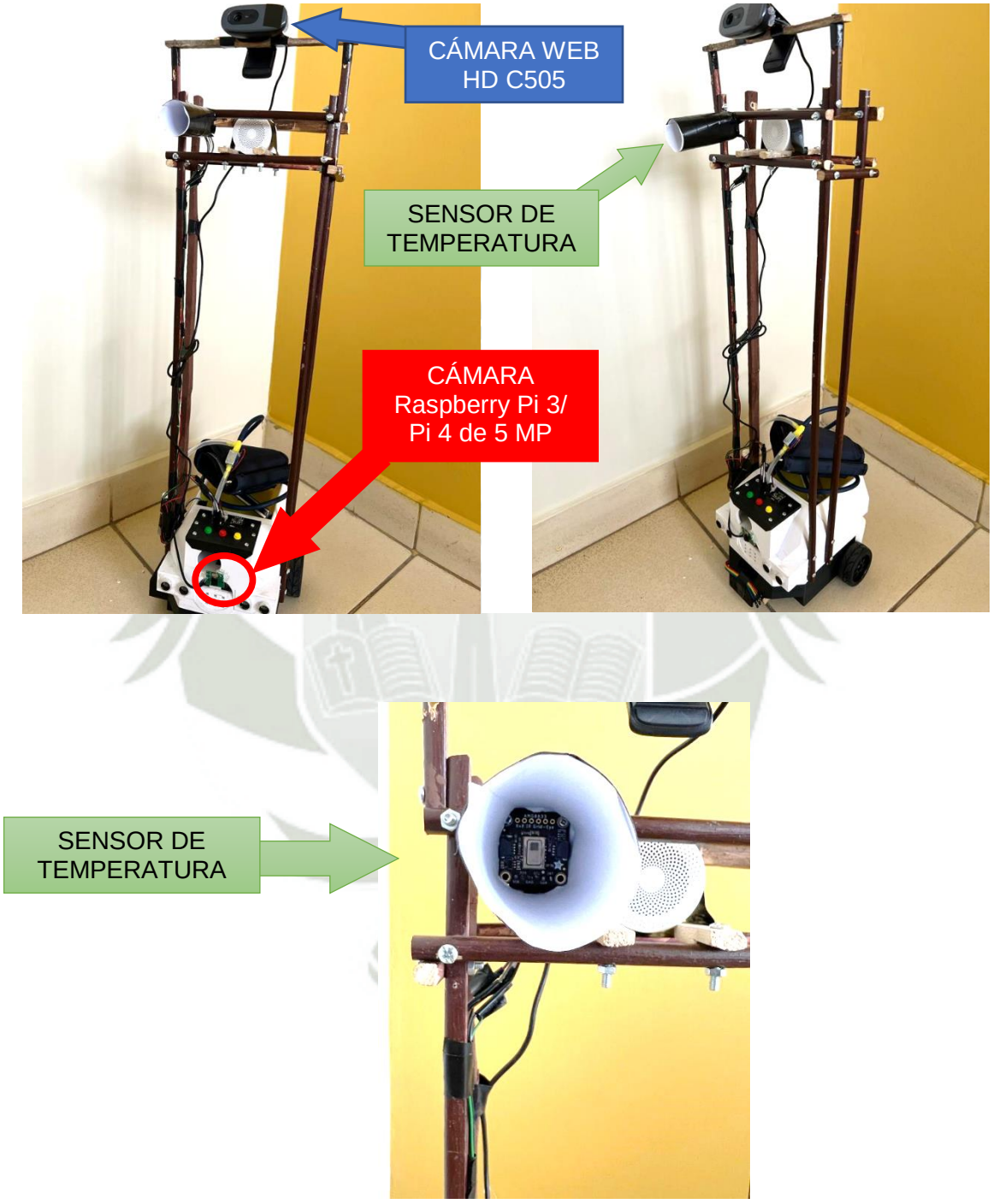


Figura 62. *Mejoras en hardware*

Fuente: Elaboración Propia.

4.6. Rediseño de Software

4.6.1. Algoritmo de rastreo de conos azules

Una vez que se obtiene la captura de imagen con la cámara Raspberry Pi 3/ Pi 4 de 5 MP, procedemos al procesamiento de dicha imagen y extracción del objeto a rastrear:

a) Recibimos la imagen que estamos obteniendo a través de la cámara, esta imagen inicialmente es de tamaño 640 x 480 pixeles

b) Redimensionamos la imagen a la mitad: 320 x 240 pixeles

c) Una vez que ya se tiene esta parte, hacemos la conversión a espacio de color HSV; es decir vamos a pasar una imagen en RGB (la que se está presentando en la interfaz del usuario) a una imagen HSV.

d) Después, de esta imagen vamos a extraer únicamente el canal de tonalidad (también se le conoce como Hue).

e) Posteriormente una vez que se tiene esta imagen de tonalidad se aplica un filtro de suavizado “Filtro Gauss”.

f) Para la imagen resultante, se va a aplicar una segmentación; la segmentación es binaria en donde vamos a establecer un umbral de acuerdo al color que deseamos detectar (AZUL). Entonces, mediante el análisis del Histograma de la imagen, se establece un valor de “200”.

Nota: En este caso, del espacio de color de tonalidad, detectamos el pico que corresponde al objeto o al color del objeto que queremos detectar.

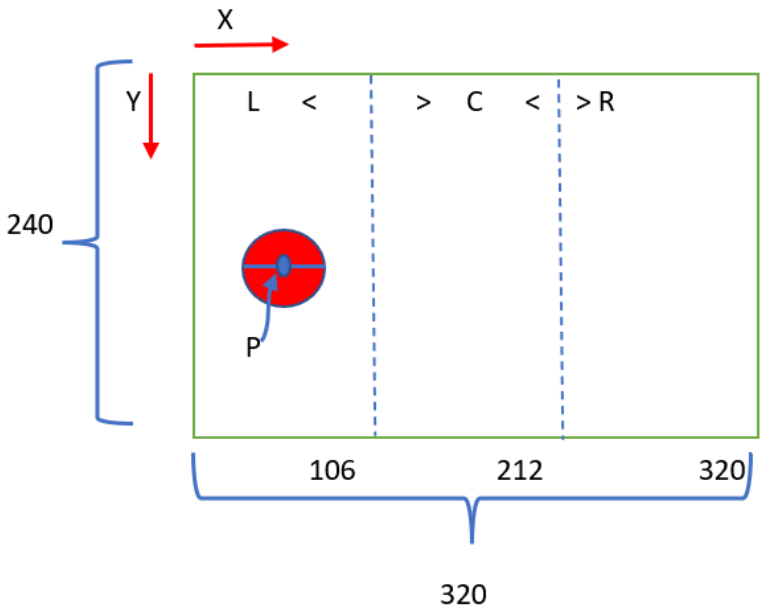
g) Una vez aplicado el umbral, obtenemos una imagen binaria, a la cual, le vamos a aplicar un Filtro Morfológico de Erosión (Kernel de 5x5, tipo circular para aplicar esta operación).

Nota: Esto lo hacemos para eliminar pequeñas partículas que puedan quedarse en la imagen, que corresponde a pequeños ruidos que pueda captar la cámara. Con esto eliminamos ese efecto.

h) Luego eliminamos los marcos de la imagen para evitar colisiones con las esquinas de objetos. Entonces, vamos a eliminar 16 pixeles de cada orilla de imagen. (eliminar básicamente es colocar estos valores como pixeles de valor “0”).

i) Posteriormente hacemos el etiquetado de la imagen resultante del paso anterior, al hacer el etiquetado vamos a poder extraer el objeto que se encuentre dentro de la imagen.

j) Luego ubicamos las coordenadas del objeto.



Aquí tenemos la representación de la imagen con el tamaño de 240 x 320; en el eje "Y" vamos a tener 240 pixeles que son filas, y en columnas que es el eje "X" tenemos 320 pixeles, esta imagen la vamos a dividir en tres columnas.

Cada división se hace del mismo tamaño, entonces la parte izquierda corresponde a las coordenadas de 0 a 106 en el eje "X", la parte central de 106 a 212 y la parte derecha de 212 a 320.

Tomando como ejemplo, que el objeto se encuentre en la parte izquierda, obtenemos su coordenada mínima del eje "X" y la coordenada máxima, para poder hacer una diferencia de esto y obtener el punto central del objeto que estamos detectando.

Pues obtenemos el punto central y lo comparamos para saber en qué cuadrante se encuentra. Entonces, por ejemplo, si el punto central del objeto tendría un valor aproximado a 50 en el eje "X", entonces comparamos y decimos: si el punto donde se encuentra el objeto de interés es menor que 106, entonces se encuentra en la parte izquierda de la imagen, y si tuviera un valor de 106 a 212 debe estar en la parte central, caso contrario, si estaría de 212 a 320 estaría en la parte derecha. Es así como encontramos la ubicación del objeto, el cual, según nuestro ejemplo, se encontraría en la parte izquierda.

Por lo tanto, enviaría el comando a la tarjeta ATMega328 para que active los motores y gire a la parte izquierda, para tratar de posicionar el objeto en la parte central. Y una vez que se encuentre en la parte central, el robot va a avanzar hacia adelante tratando de alcanzar el objeto.

Ahora, si detecta un obstáculo, se activará la cámara térmica y comenzará a tomar temperatura. Esto con el fin de saber si dicho obstáculo es un objeto inerte o si es una persona.

Si el robot determina que el obstáculo es una cama vacía, lo evitará y continuará su camino. Por otro lado, si el robot determina que hay una persona en la cama, se detendrá e iniciará la evaluación del paciente; luego de esto, continuará su camino.

4.6.2. Cálculos de procesamiento de imagen para reconocimiento de conos azules.

- **Conversión RGB a HSV**

Sea *MAX* el valor máximo de los componentes (*R*, *G*, *B*), y *MIN* el valor mínimo de esos mismos valores, los componentes del espacio HSV se pueden calcular como:

$$H = \begin{cases} \text{no definido}, & \text{si } MAX = MIN \\ 60° \times \frac{G-B}{MAX-MIN} + 0°, & \text{si } MAX = R \text{ y } G \geq B \\ 60° \times \frac{G-B}{MAX-MIN} + 360°, & \text{si } MAX = R \text{ y } G < B \\ 60° \times \frac{B-R}{MAX-MIN} + 120°, & \text{si } MAX = G \\ 60° \times \frac{R-G}{MAX-MIN} + 240°, & \text{si } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{si } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{en otro caso} \end{cases}$$

$$V = MAX$$

- **Filtro de suavizado GAUSS**

Un filtro gaussiano es un filtro de paso bajo que se utiliza para reducir el ruido (componentes de alta frecuencia) y las regiones borrosas de una imagen.

En el proceso de usar el Filtro Gaussiano en una imagen, primero definimos el tamaño del Kernel / Matrix que se usaría para eliminar la imagen. Los tamaños son generalmente números impares, es decir, los resultados generales se pueden calcular en el píxel central. Además, los núcleos son simétricos y, por lo tanto, tienen el mismo número de filas y columnas. Los valores dentro del Kernel son calculados por la función gaussiana, que es la siguiente:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Función gaussiana bidimensional dónde,

x → valor de la coordenada X

y → valor de la coordenada Y

π → Constante matemática PI (valor = 3,13)

σ → Desviación estándar

Usando la función anterior, se puede calcular un Kernel gaussiano de cualquier tamaño, proporcionándole los valores apropiados. Una aproximación del núcleo gaussiano de 3 × 3(bidimensional) con desviación estándar = 1, aparece de la siguiente manera

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- **Histograma**.

Dada la representación digital de una imagen por medio del arreglo de N filas por M columnas se determina una matriz M×N, en la cual la representación digital de bitmap estará dada por la función distribución f (m, n), para n∈ [0, N−1] y m∈ [0, M−1], típicamente N y M son potencias de 2, como ya se enunció.

El histograma de una imagen h(i), comúnmente denominado *"image enhancenment"* o *"image characterization"* es un vector que da cuenta de la cantidad de *pixels* dentro de la imagen con un cierto valor de elemento. Es decir, para una imagen de α-bits, se tiene:

$$h(i) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \delta(f(n,m) - i) \forall i \ \in [0, 2^{\propto} - 1]$$

- **Filtro morfológico de erosión**

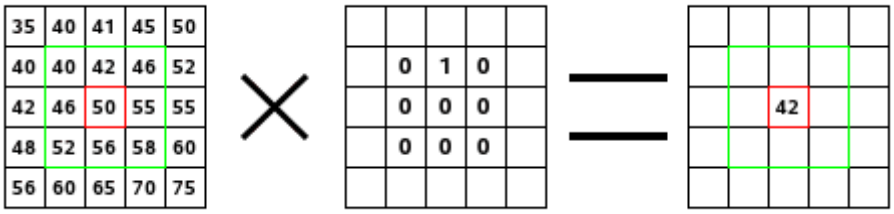
Erosión es el tratamiento de una matriz por otra que se llama "Kernel".

El filtro matriz de erosión usa una primera matriz que es la imagen que será tratada. La imagen es una colección bidimensional de píxeles en coordenada rectangular. El Kernel usado depende del efecto deseado.

El GIMP usa matrices 5x5 o 3x3. Consideraremos solo las matrices 3x3, son las más usadas y son suficiente para los efectos deseados. Si todos los valores de un Kernel se seleccionan a cero, el sistema la considerará como una matriz 3x3.

El filtro examina, sucesivamente, cada píxel de la imagen. Para cada uno de ellos, que llamaremos “píxeles iniciales”, se multiplica el valor de este píxel y el valor de los 8 circundantes por el valor correspondiente del Kernel. Entonces se añade el resultado, y el píxel inicial se regula en este valor resultante final.

Un ejemplo simple:



A la izquierda, la imagen de la matriz: cada píxel está marcado con su valor. El píxel inicial tiene un borde rojo. El área de acción del Kernel tiene un borde verde. En el medio, el Kernel, y a la derecha, el resultado de erosión.

4.6.3. Algoritmo de reconocimiento facial de viola-jones

Este proceso se realiza por captura de imagen con la cámara web HD C505.

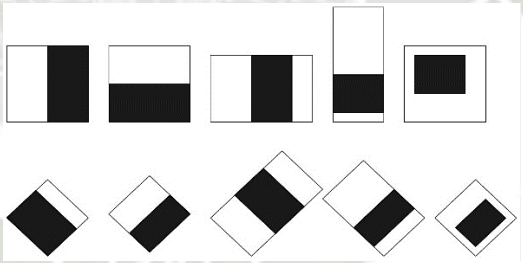
El algoritmo de viola-jones es un algoritmo de detección de rostros con un coste computacional bajo que permite que sea empleado en tiempo real, propuesto por Paul Viola, de Mitsubishi Electric Research Labs, y Michael Jones, de Compaq CRL, en julio de 2001.Su desarrollo fue motivado por el problema de la detección de caras, donde sigue siendo ampliamente utilizado, pero puede aplicarse a otras clases de objetos que, como las caras, estén caracterizados por patrones típicos de iluminación.

El algoritmo se basa en la comparación entre las intensidades luminosas de regiones rectangulares de las imágenes denominadas Características Haar-Like que calcula empleando una imagen integral. Estos clasificadores, que por sí mismos tienen una probabilidad de acertar solo ligeramente superior a la del azar, se agrupan en una cascada empleando un algoritmo de aprendizaje basado en AdaBoost para conseguir un alto rendimiento en la detección, así como una alta capacidad discriminativa en las primeras etapas.

4.6.4. Cálculos de procesamiento de imagen para reconocimiento facial

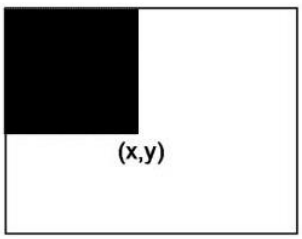
- **CARACTERISTICAS HAAR LIKE**

Haar-Like es una característica simple rectangular que se usa como una característica de entrada para el clasificador en cascada. En La Figura a continuación, hay algunos filtros basados en la característica Haar-Like. Aplicando cada uno de estos filtros en uno. En el área especial de la imagen, las sumas de pixeles debajo de las áreas blancas se restan de las sumas de pixeles debajo de las áreas negras. Es decir, el peso del área blanca y negra se puede considerar como ”1” y “-1”, respectivamente.



- **Imagen Integral**

Imagen integral es un método rápido para calcular la característica Haar-Like. La imagen integral es la suma de todos los valores de pixeles en la parte superior e izquierda de la posición (x, y). El área negra de la siguiente figura se muestra la imagen integral.



La localización x, y, contiene la suma de los pixeles de la parte superior izquierda de la imagen y se puede calcular como se indica a continuación

$$II(x,y) = \sum_{x' \leq x, y \leq y} Im(x', y')$$

Donde $II(x,y)$ es la imagen integral e $Im(x^{\wedge\prime}, y^{\wedge\prime})$ es la imagen original.

- **ALGORITMO AdaBoost**

El algoritmo de AdaBoost (un meta-algoritmo de machine learning) para elegir características y mejorar el rendimiento se usa repetidamente AdaBoost es un método de clasificación que combina varios clasificadores básicos para formar un único clasificador más complejo y preciso.

- Dado un conjunto de imágenes $(x_i, y_i), \ldots\ldots, (x_n, y_n)$ $donde\ y_i = 0{,}1$ para muestras negativas y positivas respectivamente.
- Inicializar los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0{,}1$, donde m es el numero muestras negativas y l es el número de muestras positivas.
- Para t=1…, T:
  1. Normalizar los pesos

     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

     Donde w_t es la distribución de probabilidad Por cada característica j, entrena un clasificador h_j el cual es restringido por una simple característica. El error es evaluado con respecto a w_t

     $$\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

  2. Se escoge el clasificador con menor error
  3. Se actualizan los pesos

     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

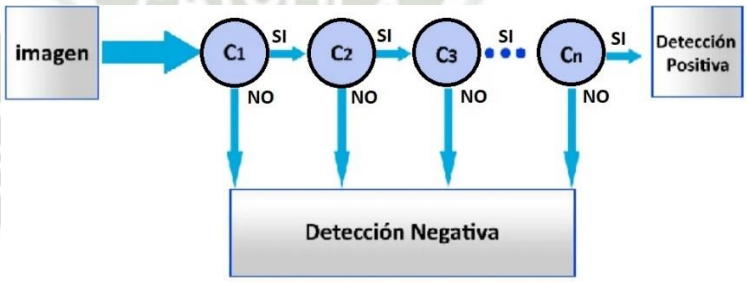
  Donde e_i = 0 si el ejemplo x_i se clasifica correctamente y 1 en caso contrario y
- El clasificador final es

$$h(x) = \begin{cases} 1\ si\ \sum_{t=1}^{T} \propto_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \propto_t \\ 0\ en\ caso\ contrario \end{cases}$$

donde $\propto_t = \frac{1}{\beta_t}$

- **CLASIFICADOR EN CASCADA**

En vez de construir un único clasificador mediante el proceso AdaBoost, se pueden construir clasificadores más pequeños y eficientes que rechacen muchas ventanas negativas (es decir, aquellas que no incluyan ninguna instancia del objeto buscado) manteniendo casi todas las positivas (es decir, las que contienen una instancia del objeto buscado). Estos clasificadores más simples se utilizan para rechazar la mayoría de las ventanas de búsqueda y solo en aquellas en las que hay mayores probabilidades de encontrar caras se llama a clasificadores más complejos que disminuyan el número de falsos positivos.



4.7. Interfaz del usuario

El desarrollo de la interfaz de usuario (GUI), se llevó a cabo en el lenguaje de programación Python 3, mismo que, ya se encuentra pre - instalado en el sistema operativo Raspbian. El entorno de desarrollo (IDE) empleado fue *Thonny*. Para desarrollar la GUI, se utilizó la biblioteca *Tkinter*, ésta es nativa de Python, y permite generar interfaces de usuario mediante el uso de comandos pre - definidos. Además de *Tkinter*, también fueron necesarias otras bibliotecas, tales como:

- Time: Para establecer tiempos de retardo en la ejecución de líneas de código.
- OpenCV: Para obtener la captura de vídeo.
- Numpy: Para trabajar con estructuras de datos (Arrays).
- Pandas: Para generar archivos Excel y CSV.
- Rpi GPIO: Para hacer uso de los puertos I/O, de la tarjeta Raspberry Pi 4.
- CircuitPython: Para obtener lecturas del sensor AMG8833
- PyGame: Para la reproducción de audio en MP3.

- PILLOW: Para convertir Arrays de imágenes, a formato compatible con Tkinter, “ImageTk”
- Spidev: Biblioteca que permite el uso del protocolo de comunicación SPI en la tarjeta Raspberry Pi 4.
- Smbus: Biblioteca que permite el uso del protocolo de comunicación I2C en la tarjeta Raspberry Pi 4.
- PySerial: Biblioteca que permite la comunicación Serial en la tarjeta Raspberry Pi 4.

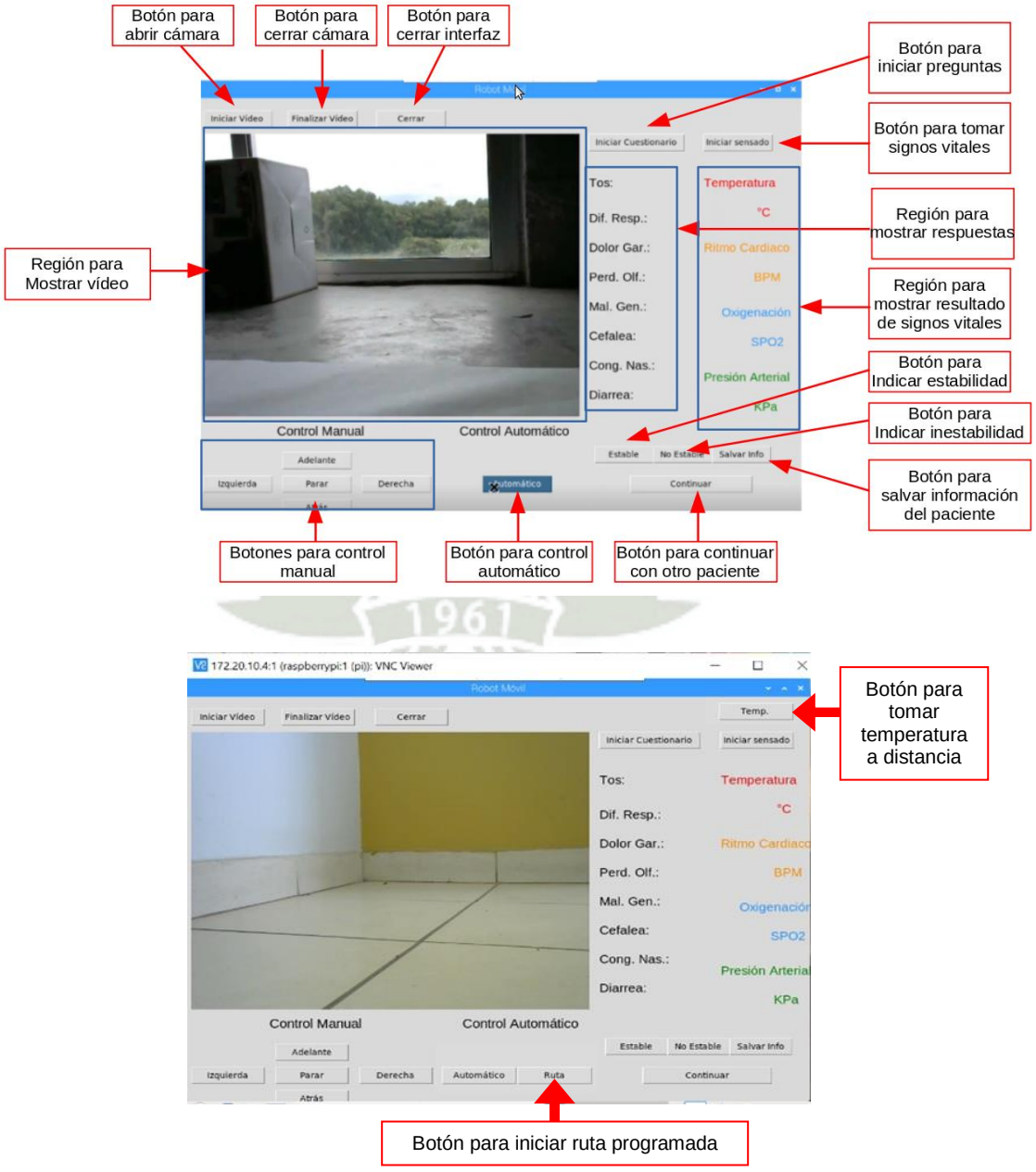


Figura 63. *Interfaz de usuario.*

Fuente: Elaboración Propia.

Nota. En la figura se muestra la interfaz de usuario, indicando una breve descripción de la funcionalidad de los diferentes botones e indicadores, que se ubican en la misma.

Este es el Programa de Python donde podremos visualizar las bibliotecas que van a ser requeridas y que serán instaladas en la tarjeta Raspberry Pi 4:

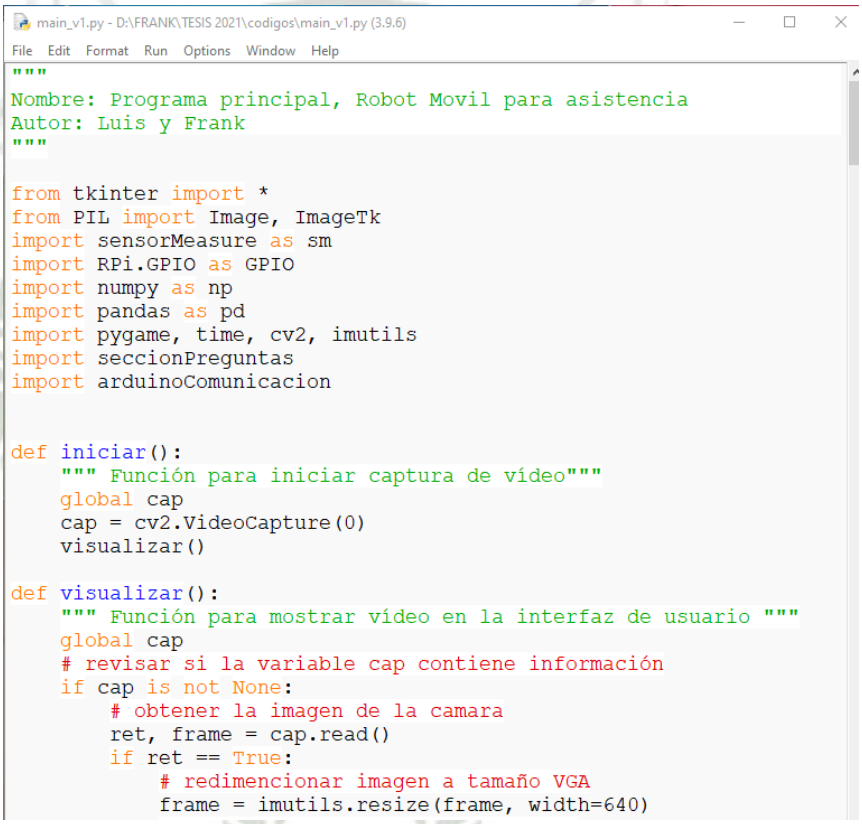


Figura 64. *Programa de Python.*

Fuente: Elaboración Propia.

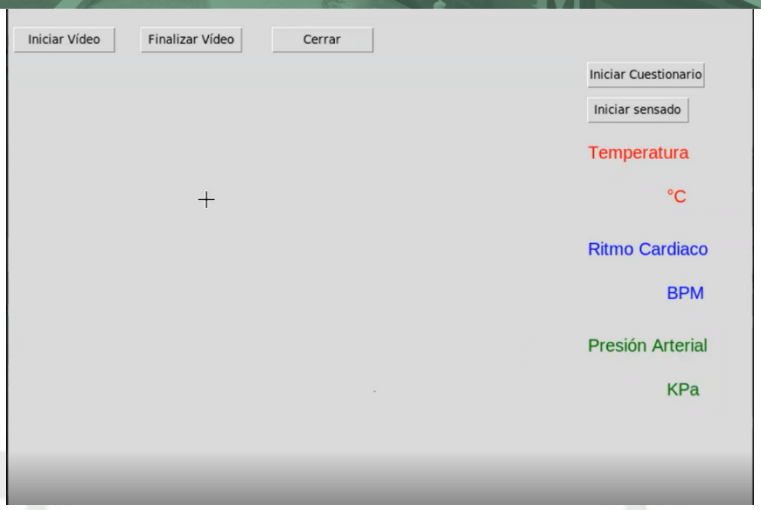


Figura 65. *Interfaz de botones superiores.*

Fuente: Elaboración Propia.

Nota. En la figura, Podemos encontrar el de Iniciar vídeo, finalizar vídeo y cerrar.

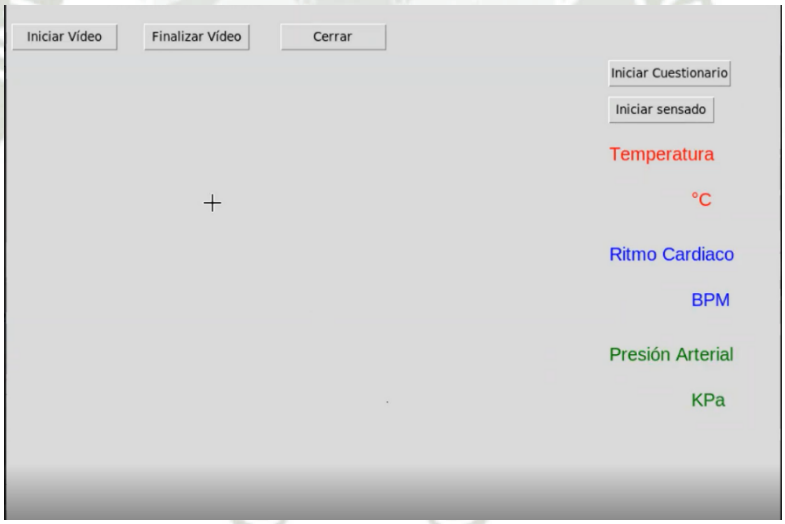


Figura 66. *Interfaz de pulsadores derechos.*

Fuente: Elaboración Propia.

Asimismo, en la figura encontramos en la interfaz de pulsadores derechos, “iniciar cuestionario” e “Iniciar Sensado”, que tienen la siguiente función:

- Iniciar Cuestionario: Este hace referencia a las preguntas por síntomas que se le serán enviadas a través del altavoz al usuario y que serán contestadas por medio de los botones que se encuentran en el robot.

- Iniciar Sensado: Este ya comenzara con la rutina para obtener la lectura de temperatura (°C), ritmo cardíaco (BPM), saturación de oxígeno (SPO2) y presión arterial (KPa).

- Posteriormente se añade el botón “Temp.”, el cual también permite obtener la lectura de temperatura (en caso de que el paciente se encuentre inconsciente.)

En la parte del sensado, comprobamos que los sensores de ritmo cardíaco y temperatura (ejemplo de temperatura ambiente) funcionan. Solo falta la presión arterial, puesto que es más complejo.

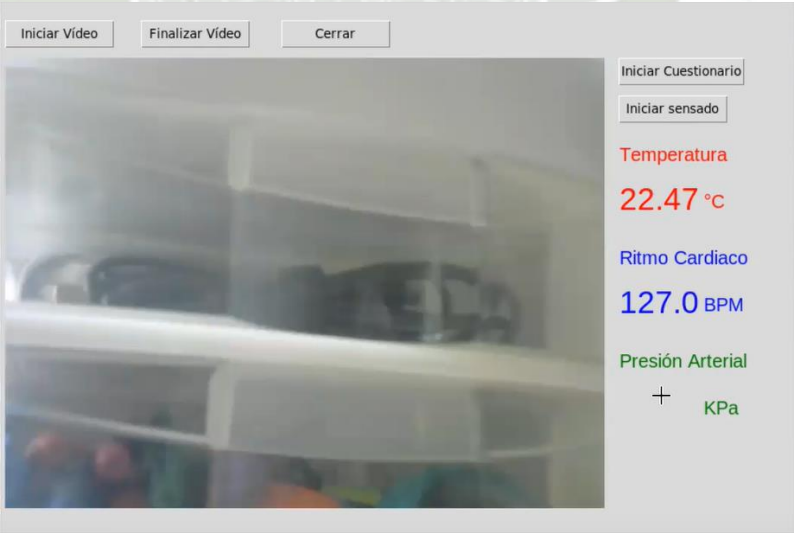


Figura 67. *Primeros resultados del robot.*

Fuente: Elaboración Propia

4.7.1. Diagramas de flujo

A) Diagrama de flujo del programa principal

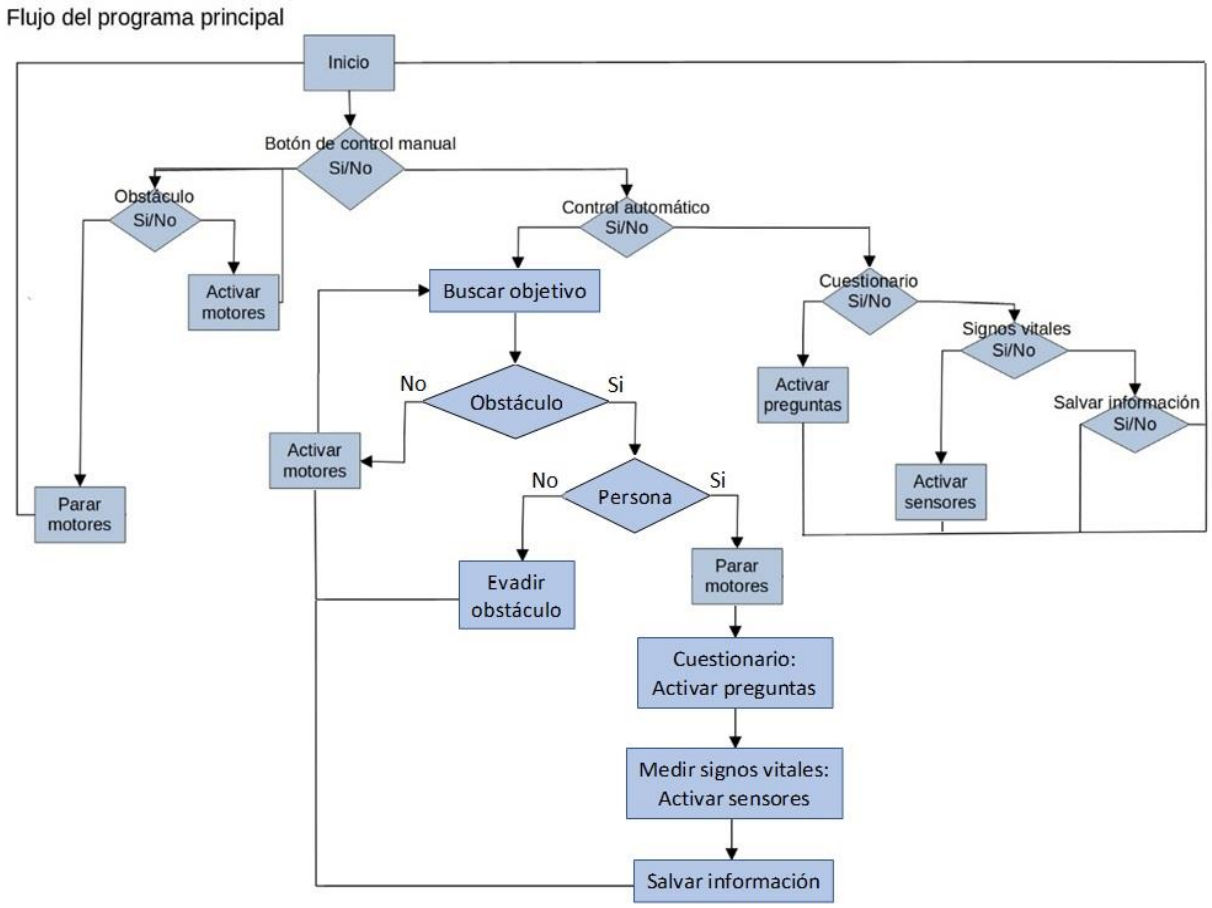
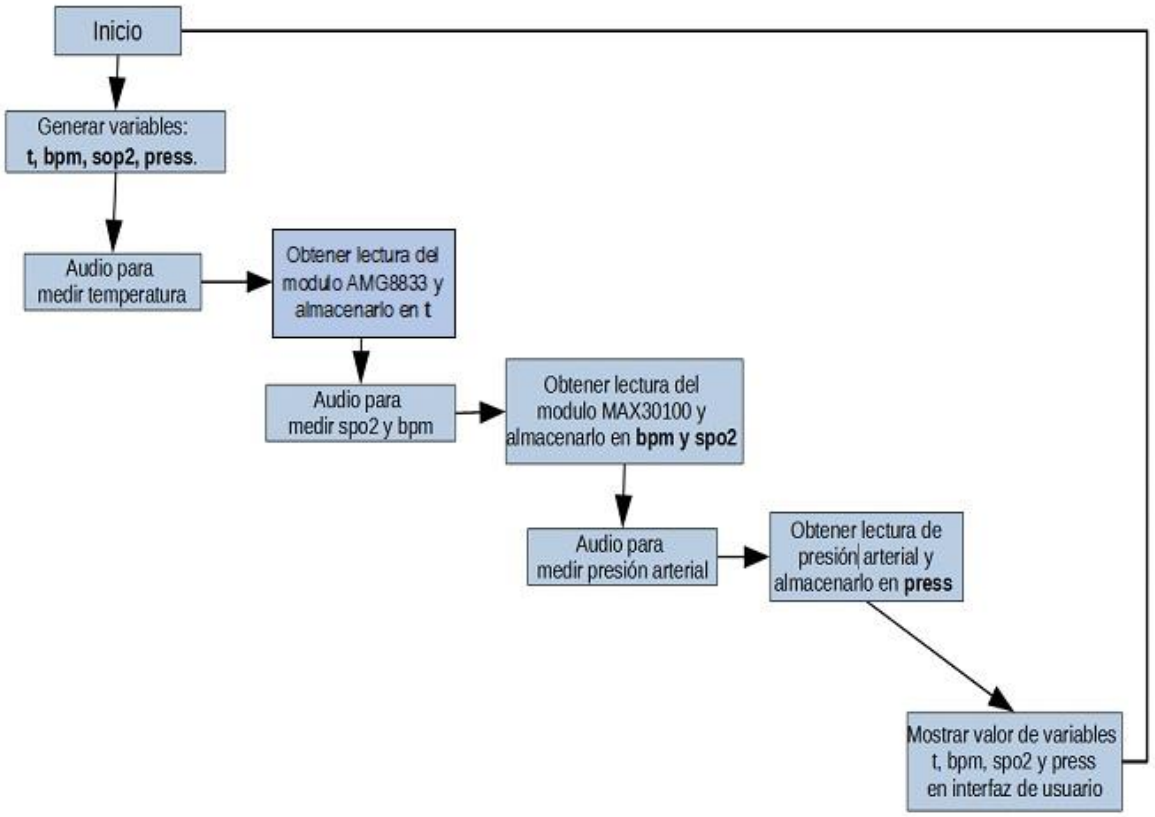


Figura 68. *Diagrama de flujo del programa principal.*

Fuente: Elaboración Propia.

B) Subrutina de toma de signos vitales: temperatura, ritmo cardiaco, oxigenación y presión arterial

Para la toma de signos vitales también se hace uso de una serie de audios, los cuales permitirán guiar al paciente para ejecutar una acción para tomar así sus signos vitales, mediante los diferentes sensores incorporados al robot.



F**igura 69**. *Diagrama de flujo del programa para obtener signos vitales.*

Fuente: Elaboración Propia.

El programa comienza indicándole al paciente que se llevará a cabo la toma de sus signos vitales (bloque inicio).

Posteriormente se generan cuatro variables de nombre: t: para almacenar el valor de temperatura; bpm: para almacenar el valor del ritmo cardíaco; spo2: para almacenar el valor de oxigenación; y press: para almacenar el valor de presión arterial (bloque generar variables).

A continuación, se reproduce el siguiente audio (bloque audio para medir temperatura): *Vamos a comenzar con la toma de temperatura, por favor coloque su mano frente al sensor sin tocarlo.*

Una vez que se concluye la reproducción del audio, se procede a medir la temperatura empleando el módulo AMG8833 (sensor infrarrojo), y ésta se almacena en la variable t.

En seguida se continúa con la reproducción del audio para medir oxigenación y ritmo cardíaco, el audio menciona lo siguiente: *Ahora voy a tomar su ritmo cardíaco, por favor coloque su dedo índice sobre el sensor, mantenga hasta que se le indique.*

Una vez concluido lo anterior, se da lugar a la toma de presión arterial; para esto inicialmente se reproduce el siguiente audio*: Por último, voy a tomar su presión arterial, por favor colóquese el tensiómetro sobre su brazo derecho y espere.*

Una vez concluida la reproducción de audio, se da lugar la toma de presión arterial; en este punto se emplea el programa “presión arterial” (que se presenta en el siguiente diagrama). El programa regresa el valor la presión obtenida del sensor MPX10DP en KPa y éste es almacenado en la variable press.

Por último, se indica al paciente que se retire el tensiómetro, y que con esto ha terminado el proceso de toma de signos vitales; resta esperar el dictamen del médico.

Con lo anterior, el valor de las variables se muestra en la interfaz, para que pueda ser visualizado por el médico o personal de salud.

C) Subrutina de toma de temperatura:

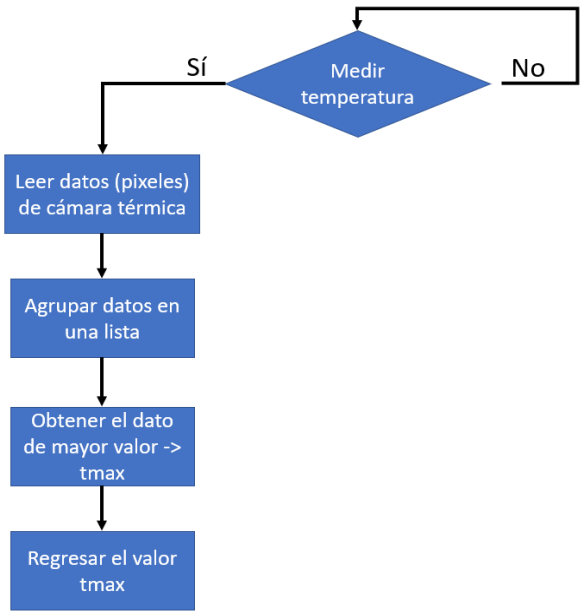


Figura 70. *Diagrama de flujo del programa para temperatura*

Fuente: Elaboración Propia.

D) Subrutina de toma de presión arterial

En el siguiente diagrama se presenta la secuencia diseñada para medir la presión arterial.

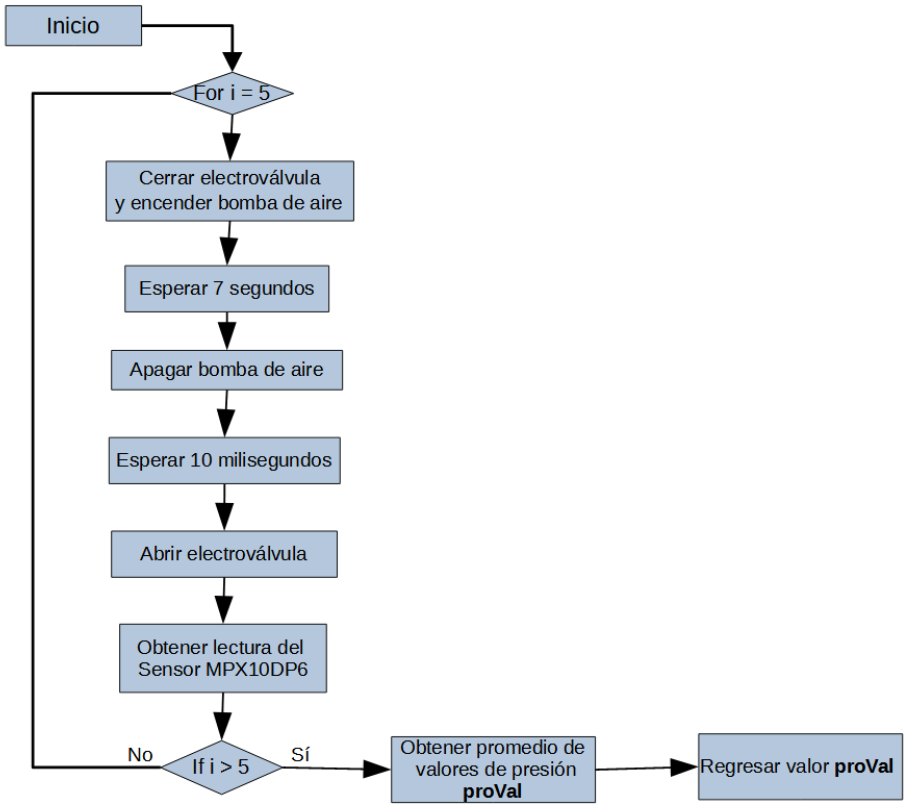


Figura 71. Diagrama de flujo del programa para obtener presión arterial.

Fuente: Elaboración Propia.

El sistema para obtener la presión arterial está compuesto de los siguientes elementos, mismo que se presentan en la Figura 72:

- Bomba de aire.
- Cople de tres, que permite la unión entre la bomba, electroválvula y brazalete.
- Electroválvula (Normalmente abierta).
- Sensor de presión de aire (MPX10DP).
- Brazalete.

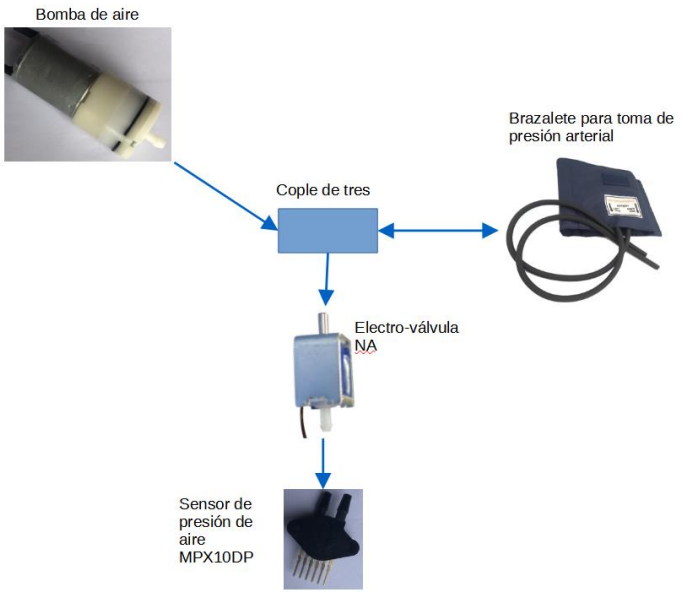


Figura 72. Sistema para obtener la presión arterial.

Fuente: Elaboración Propia.

Como se puede observar en el diagrama del programa, el proceso de obtención de presión arterial consiste básicamente de dos procesos; el primero consiste en producir presión de aire y dirigirla hacia el brazalete, esto se hace mediante el encendido de la bomba de aire, excitación de la electroválvula (cerrar válvula) para evitar paso de aire hacia el sensor de presión. Este proceso se mantiene en operación durante siete segundos, el mismo que se representa en la Figura 73, donde las líneas rojas representan el paso de aire.

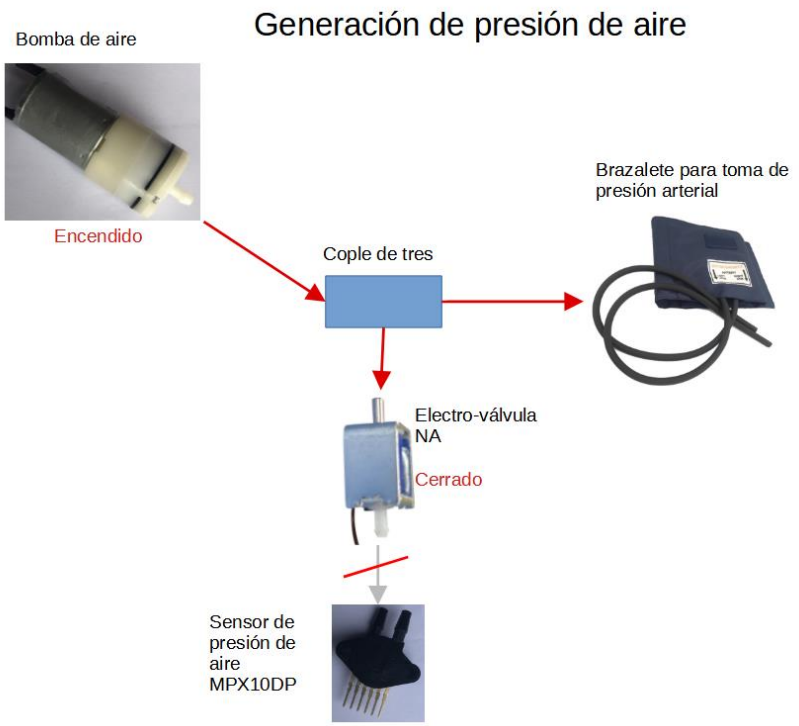


Figura 73. Generación de presión de aire.

Fuente: Elaboración Propia.

Una vez concluido el proceso anterior, se apaga la bomba de aire y se mantiene un tiempo de espera por diez milisegundos, y posteriormente se da comienzo a la obtención de la presión arterial, esto se hace mediante la apertura de la electroválvula, para permitir el flujo de aire hacia el sensor de presión. Esta operación se presenta en la siguiente imagen. Donde las flechas en color rojo representan el flujo de aire, el cual va desde el brazalete hacia el sensor de presión.

Figura 74. Obtención de la presión arterial.

Fuente: Elaboración Propia

Los dos procesos descritos previamente se repiten cinco veces, siendo que, cada iteración almacena el valor de presión obtenido, y al concluir las cinco iteraciones se obtiene el promedio de los cinco valores generados; este valor se almacena en la variable “proVal”, y es la que se regresa al proceso de obtención de signos vitales para su posterior despliegue en la interfaz de usuario.

E) Subrutina de autonomía de desplazamiento

Para detectar obstáculos se obtiene la lectura de distancia medida por los sensores ultrasónicos, mismos que se encuentran en la parte frontal y trasera del robot, como se presenta en la Figura 75.

Figura 75. Parte trasera, lateral y frontal del robot.

Fuente: Elaboración Propia

A continuación, veremos el diagrama de flujo correspondiente:

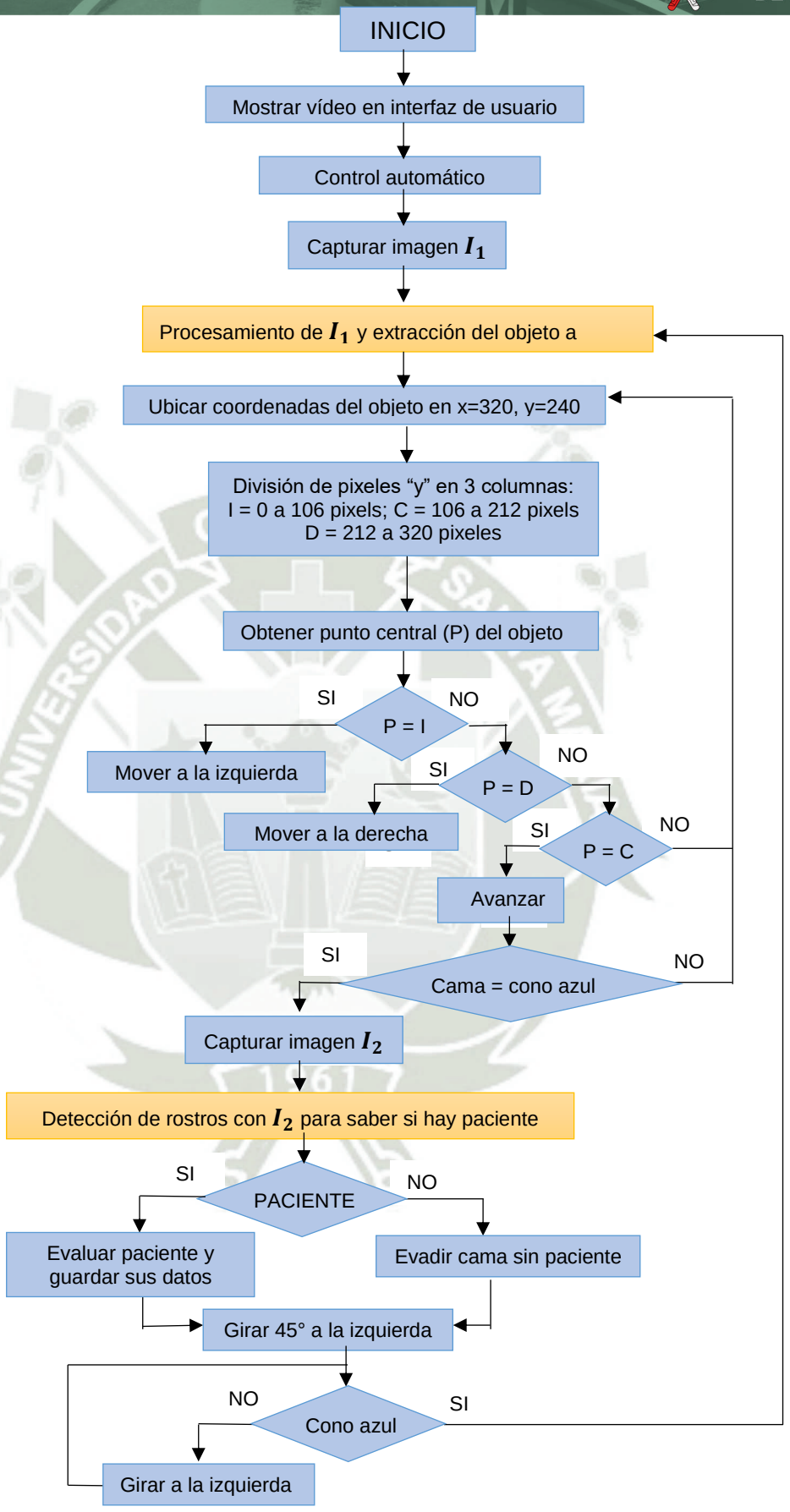


Figura 76. Diagrama de flujo del programa para detectar obstáculos y desplazamiento.

Fuente: Elaboración Propia.

El programa inicia obteniendo las lecturas de distancia a través de los sensores HCSR-04, donde si la distancia obtenida es menor a 20, en los sensores de la parte frontal, entonces el robot solo se podrá mover hacia atrás, por otro lado, si los sensores de la parte trasera detectan un obstáculo, el robot solo podrá moverse hacia adelante, y en caso de que los sensores de ambas partes detecten un objeto, entonces el robot no podrá moverse.

F) Subrutina de detección de conos azules en una imagen

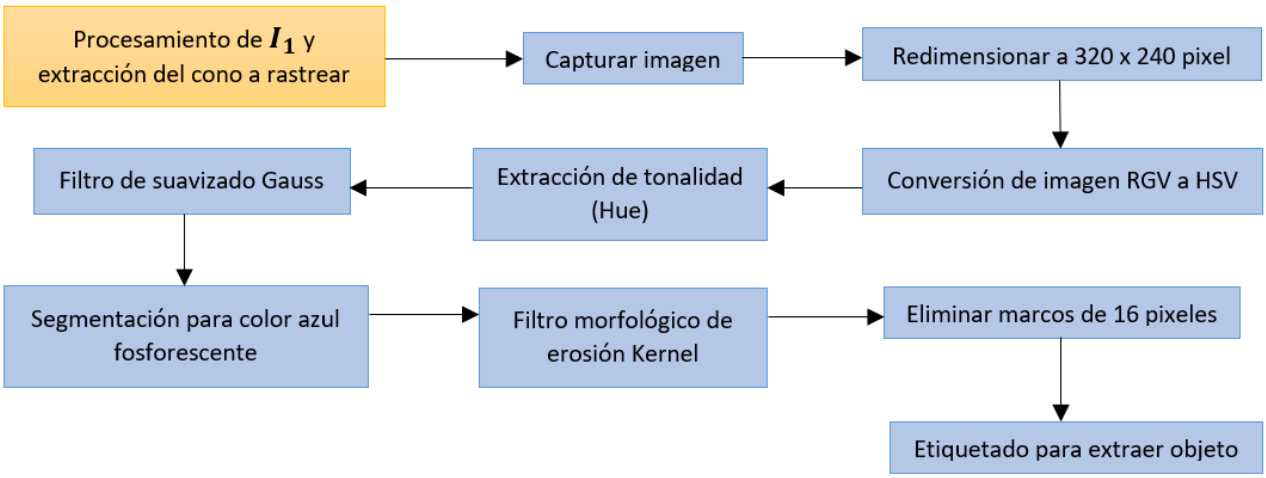
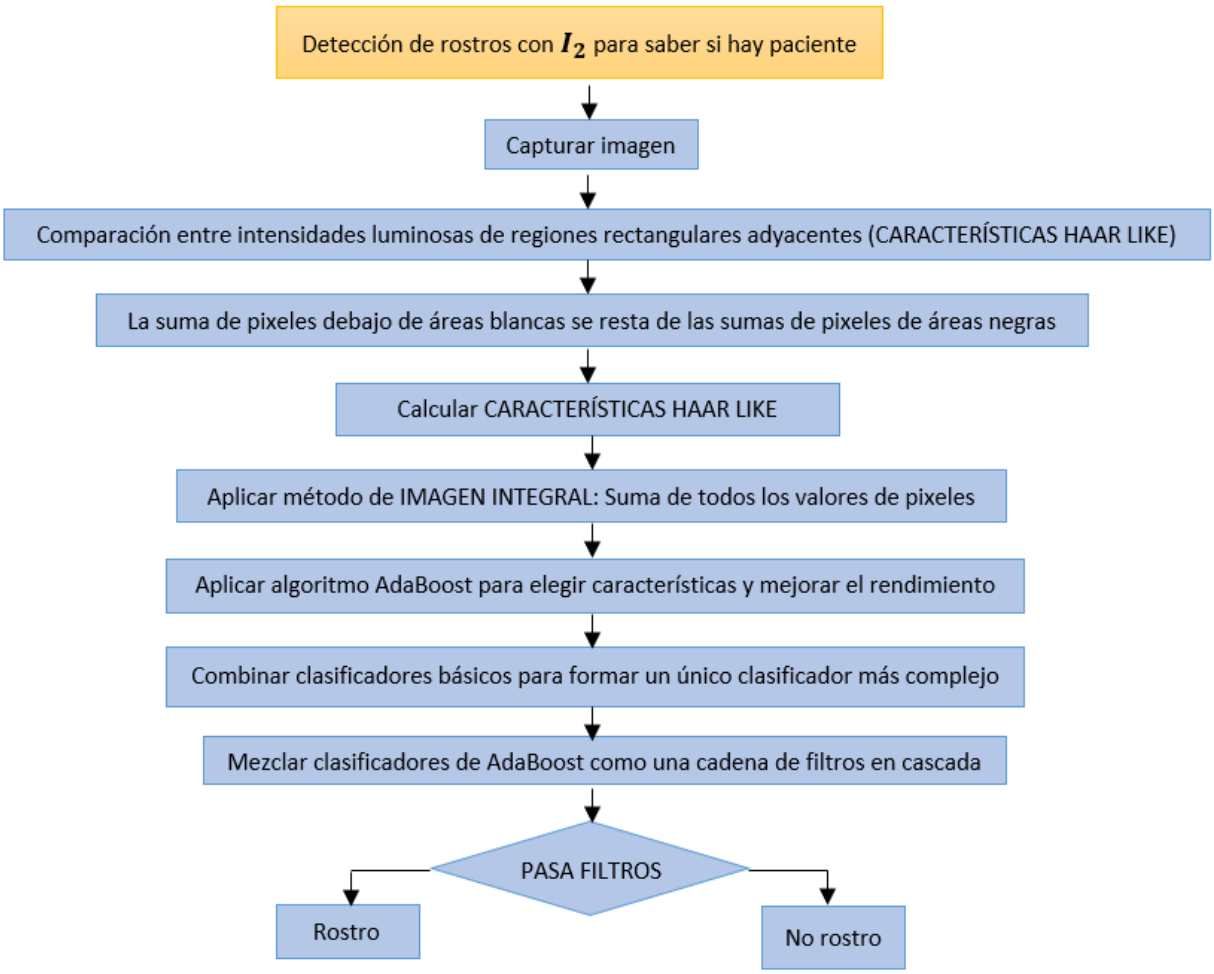


Figura 77. Diagrama de flujo del programa para detectar color azul

Fuente: Elaboración Propia.

G) Subrutina de detección de rostros con algoritmo viola jones



H) Subrutina para inicio y captura de video

Para llevar a cabo la captura de vídeo se emplea una cámara web, la cual es conectada a uno de los puertos USB de la tarjeta Raspberry Pi 4. Para acceder desde el programa a la cámara web, se hace uso de la biblioteca *OpenCV,* esta biblioteca permite la captura de fotografías y vídeo por medio de una cámara; además esta biblioteca contiene funciones para procesamiento digital de imágenes y visión por computadora, como por ejemplo la detección de colores y formas.

En el siguiente diagrama se presenta la secuencia para obtener el vídeo a través de la cámara web y mostrarla en la interfaz de usuario.

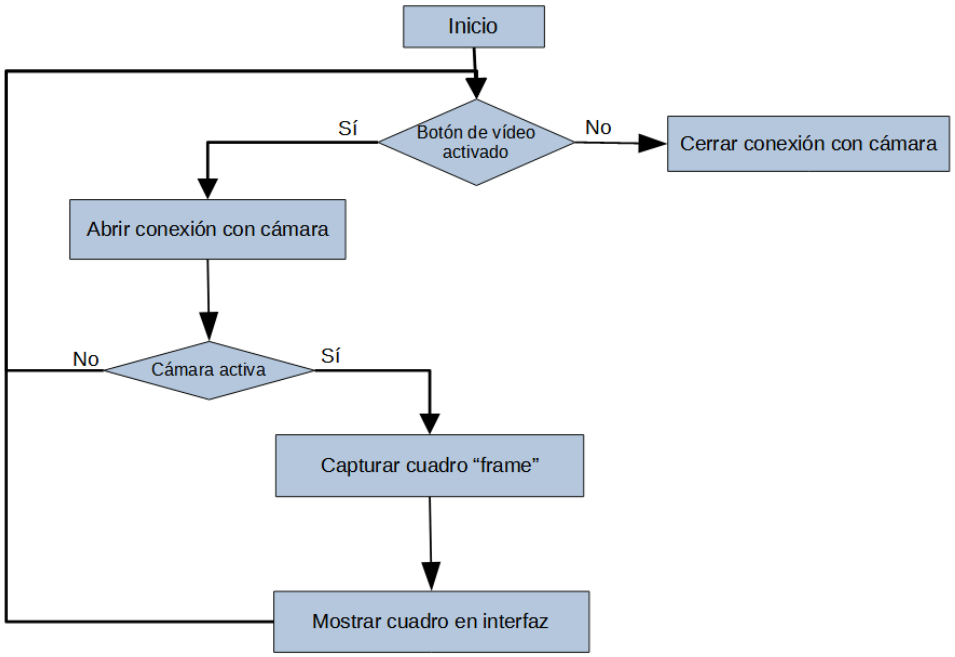


Figura 78. Diagrama de flujo de secuencia para obtener el vídeo.

Fuente: Elaboración Propia.

Este proceso comienza con la lectura del botón "iniciar vídeo", el cual se encuentra en la interfaz de usuario; si éste se encuentra desactivado, se cierra la comunicación con la cámara web, y en caso contrario, se abre la comunicación con la cámara y posteriormente se monitorea si ésta se encuentra activa, ya que pudiese darse el caso de que se encuentre desconectada; si lo anterior no ocurre, entonces se continua con la captura de un cuadro o frame, y enseguida es mostrado en la interfaz de usuario. Este proceso se repite hasta que se desactiva el botón "iniciar vídeo".

I) Subrutina para guardar la información del paciente en un archivo

Una vez que se ha llevado a cabo el cuestionario y la recolección de los signos vitales del paciente, es posible guardar esta información, para ello la interfaz de usuario incorpora un botón de nombre "salvar info", éste permite guardar los datos del paciente actual, y con el botón "continuar" posteriormente se borra los datos, limpiando el valor de las variables que almacenan las respuestas y valores de signos vitales, para continuar con un nuevo paciente.

A continuación, se presenta el proceso para almacenar la información de signos vitales y respuestas del paciente. Donde se parte de que previamente se ha recabado dicha información, empleando los diagramas expuestos previamente.

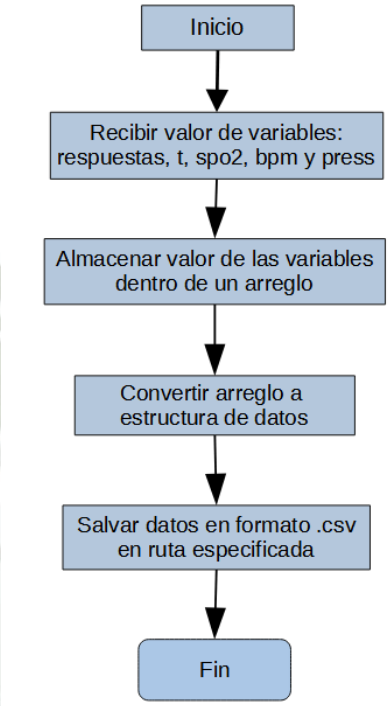


Figura 79. Diagrama de flujo del programa para guardar información del paciente en un archivo**.**

Fuente: Elaboración Propia.

El programa comienza recibiendo el valor de las variables generadas en la etapa de cuestionario y obtención de signos vitales. Posteriormente estas variables son almacenadas dentro de un arreglo y en seguida se convierte a tipo estructura de datos, esto se hace mediante el uso de la biblioteca *Pandas*, esta biblioteca permite la lectura y escritura, de archivos en diferentes formatos como Excel, Word, csv, etc.

Una vez que la información se ha convertido a tipo estructura de datos, se procede a salvarla en formato csv. La ruta en donde ésta es salvada es el escritorio, en la carpeta de nombre “archivos pacientes”.

J) Diagrama de flujo de las alarmas

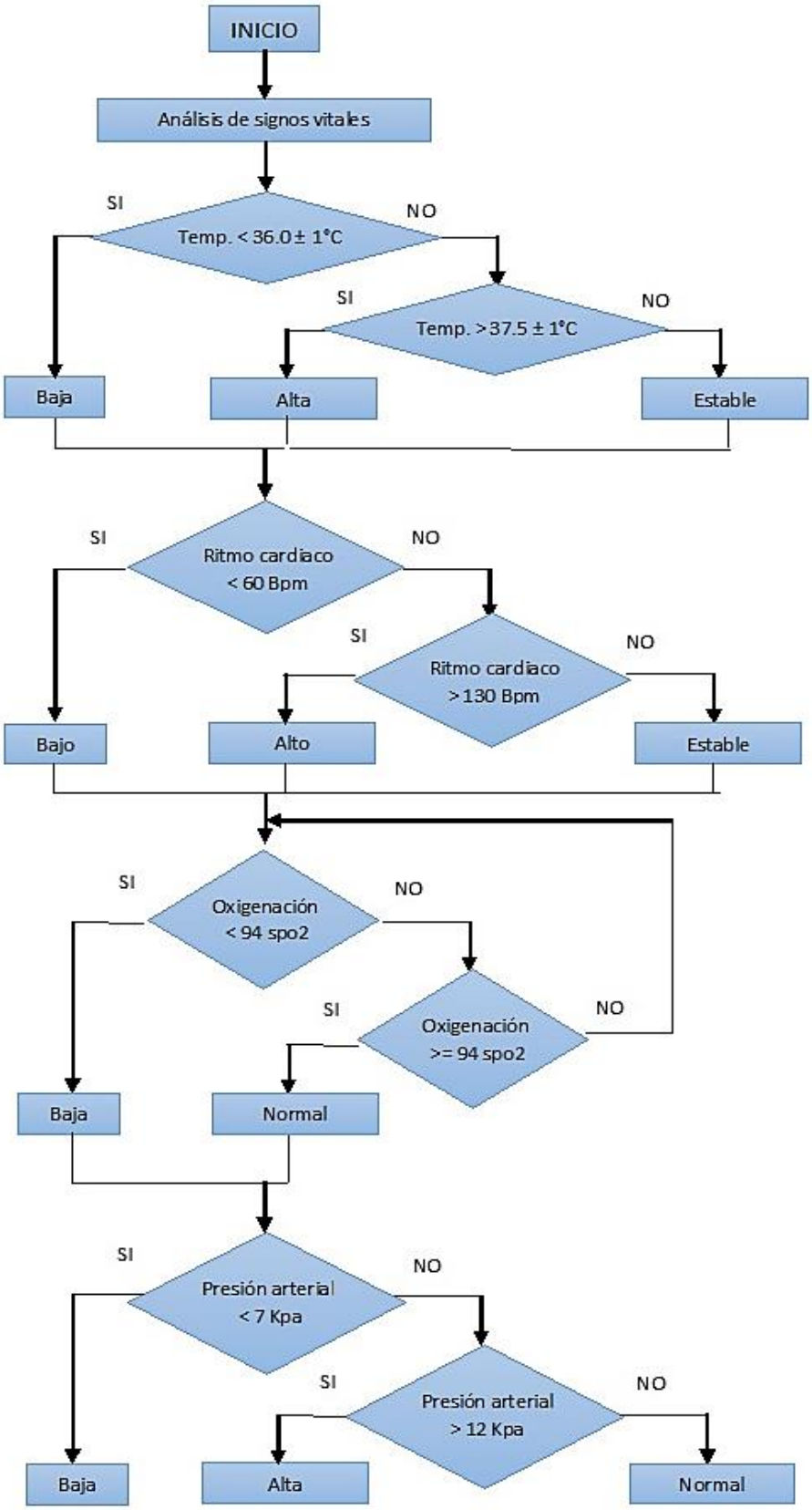


Figura 80. Diagrama de flujo del programa para funcionamiento de alarmas

Fuente: Elaboración Propia.

- **Rango para las alarmas.**

RANGO PERMITIDO		
	Mínimo	**Máximo**
Temperatura (°C)	36.0 ± 1	37.5 ± 1
Ritmo Cardíaco (BPM)	60	130
Saturación de oxígeno (SPO2)	94	100
Presión arterial (Kpa)	7	12

Nota: Se considera al sensor de temperatura un margen de error de ± 1.

Tabla 15. *Rango de alarmas*
Fuente: Elaboración propia

De acuerdo con el rango permitido para la toma de datos de cada sensor, se ha elaborado un código para que el robot móvil emita alarmas en caso de que sobrepase dichos limites:

- En caso de la temperatura, el paciente puede tener temperatura baja, normal (estable) o alta, así como se ve a continuación:

Temperatura baja < 36.0 ± 1°C

36.0 ± 1°C ≤ **Temperatura normal** ≤ 37.5 ± 1°C

Temperatura alta > 37.5 ± 1°C

- En caso de que el ritmo cardiaco del paciente sea menor a 60 BPM el robot móvil avisará que tiene ritmo cardiaco bajo. Si se obtiene un valor mayor a 130 BPM avisará que tiene ritmo cardiaco alto. Finalmente, en caso este entre 60 BPM y 130 BPM, avisará que tiene ritmo cardiaco normal.
- En caso de que la saturación de oxígeno del paciente sea menor a 94 SPO2 el robot móvil avisará que tiene saturación baja. Si se obtiene un valor mayor o igual a 94 SPO2, avisará que tiene saturación normal.
- En caso de que la presión arterial del paciente sea menor a 7 Kpa el robot móvil avisará que tiene presión baja. Si se obtiene un valor mayor a 12 Kpa avisará que tiene presión alta. Finalmente, en caso este entre 7 Kpa y 12 Kpa, avisará que tiene presión normal.

4.8. Conexión remota

Esta etapa trata sobre la conexión de la tarjeta (Raspberry Pi 4) con otra computadora, siendo el caso para Windows se deberá descargar el software “PARI-GP” o “PuTTY” para poder enlazar las vías SSH, por las vías SSH y la computadora deseada. PuTTY es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre. Disponible originalmente solo para Windows, ahora también está disponible en varias plataformas Unix, se está desarrollando la versión para Mac OS clásico, MacOS X. (Chitay Bautista, 2020, pág. 130)

En primer lugar, debemos ingresar a la interfaz del software de PuTTY. Procedemos a ingresar la IP de la tarjeta de (Raspberry Pi 4), la cual la configuramos para que sea estática como podemos ver en la “Figura 81”.

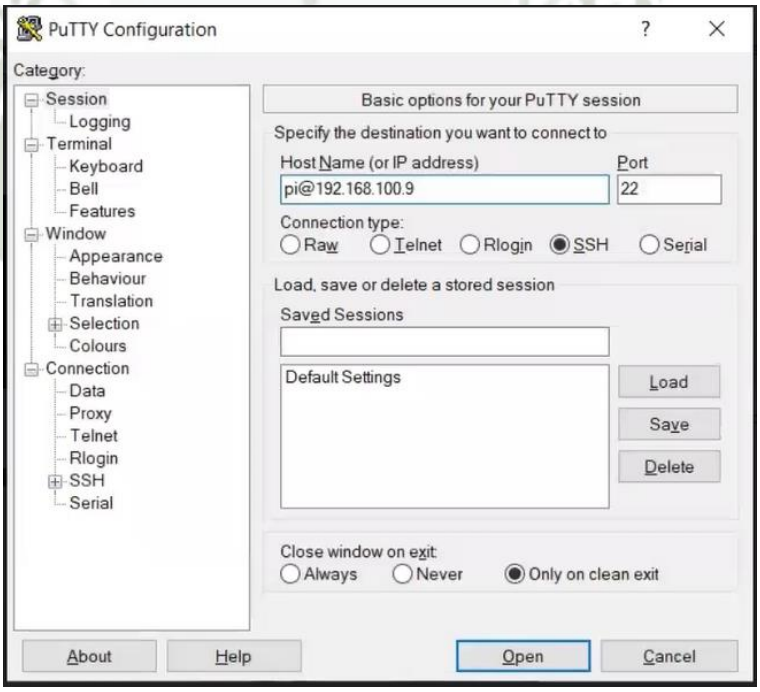


Figura 81. *Interfaz de PuTTY.*

Fuente: Elaboración Propia.

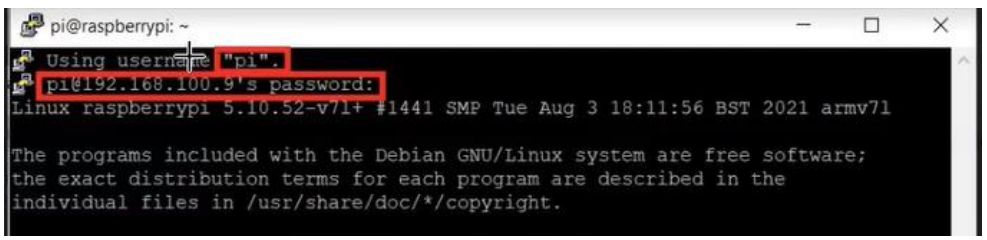


Figura 82. *Software para ingresar a la tarjeta de (Raspberry Pi 4) usuario y contraseña.*

Fuente: Elaboración Propia.

En la “Figura 82”, podemos ver que nos pide el nombre de la tarjeta de (Raspberry Pi 4), y tenemos que proceder a colocar el nombre de usuario y la contraseña, siendo el nombre “pi” (como se muestra en el primer rectángulo de marco rojo) y colocándose posteriormente la contraseña como indica el segundo rectángulo de marco rojo en la siguiente figura.

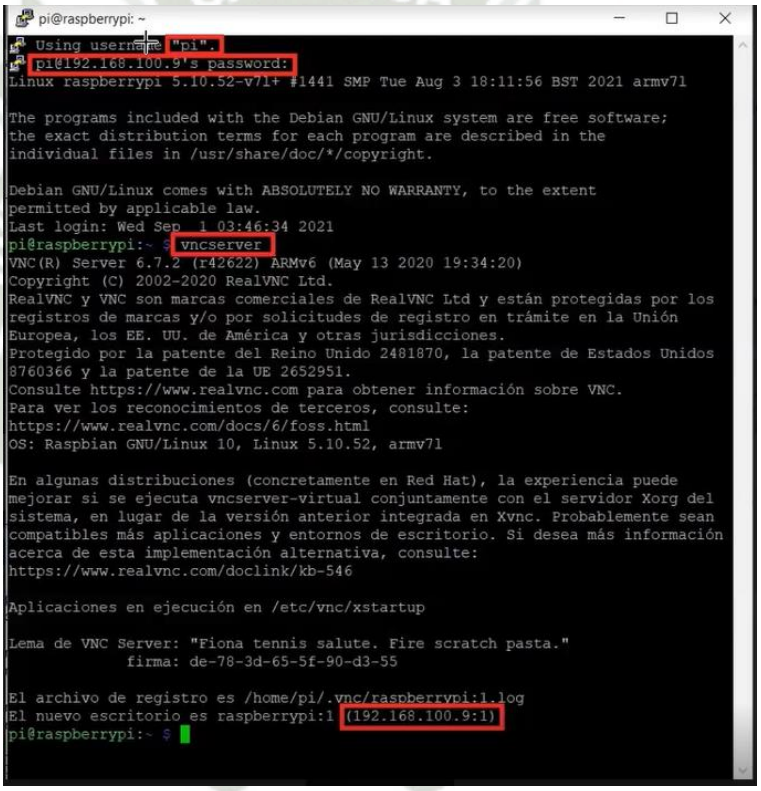


Figura 83. *Software con el acceso de la ruta para ingresar a la tarjeta (Raspberry Pi 4).*

Fuente: Elaboración Propia.

Posteriormente, al ingresar el usuario y contraseña correctos, nos da permiso a acceder a la tarjeta de (Raspberry Pi 4) por vía comando, desde una computadora Windows empezamos a controlar dicha tarjeta. Entonces, inicializamos el Servidor VNC como muestra el tercer rectángulo de marco rojo de la “Figura 83” y nos dará una ruta para poder

acceder a la tarjeta de (Raspberry Pi 4) como nos muestra el cuarto rectángulo de marco rojo de la “Figura 83”.

4.9. Ingreso a la tarjeta (Raspberry Pi 4)

VNC es un programa con el que puede hacerse cargo del control remoto de la computadora servidor a través de un cliente multiplataforma. Una vez que VNC está instalado en la computadora, se puede acceder a él desde cualquier parte del mundo a través de Internet y desde cualquier dispositivo, como una computadora o un teléfono inteligente. VNC se ofrece a sus clientes especialmente adaptado a todas las plataformas. (OSI, 2021, pág. 1)

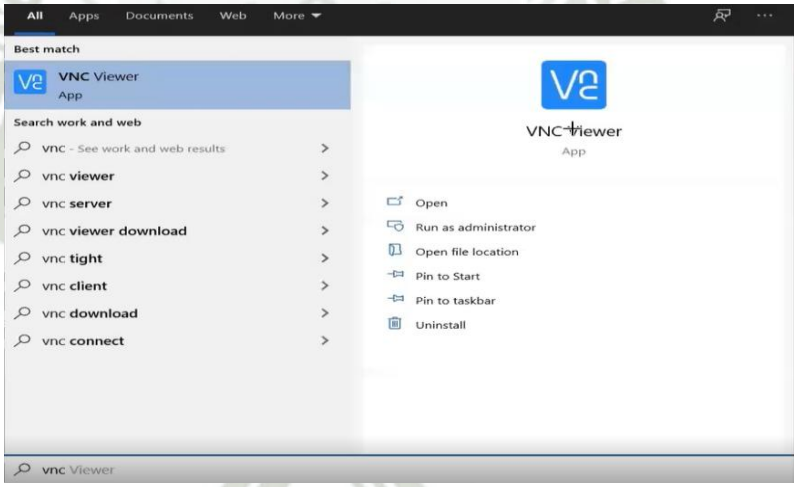


Figura 84. *Programa VNC Viewer.*

Fuente: Elaboración Propia.

Por consiguiente, ingresamos al programa VNC Viewer como vemos en la “Figura 84”, el cual es totalmente gratuito y nos servirá para observar la pantalla de la tarjeta (Raspberry Pi 4) como nos muestra en la siguiente figura.

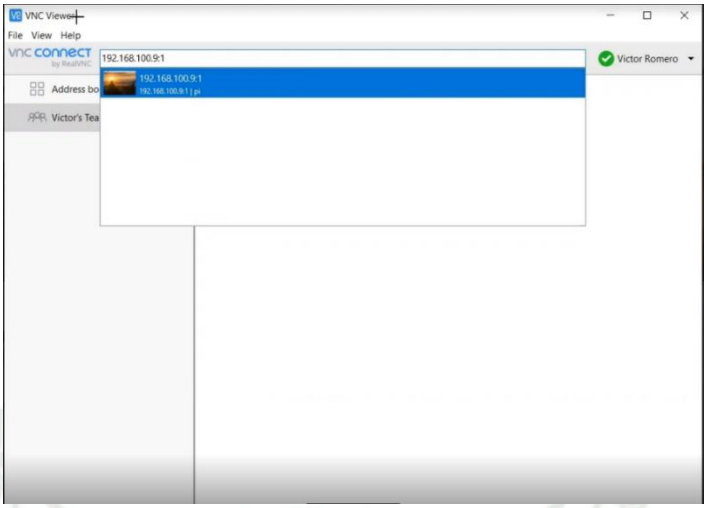


Figura 85. *Visualización de VNC Viewer.*

Fuente: Elaboración Propia.

Colocaremos el IP de la tarjeta (Raspberry Pi 4) dándonos acceso por vía remota a la tarjeta (Raspberry Pi 4).

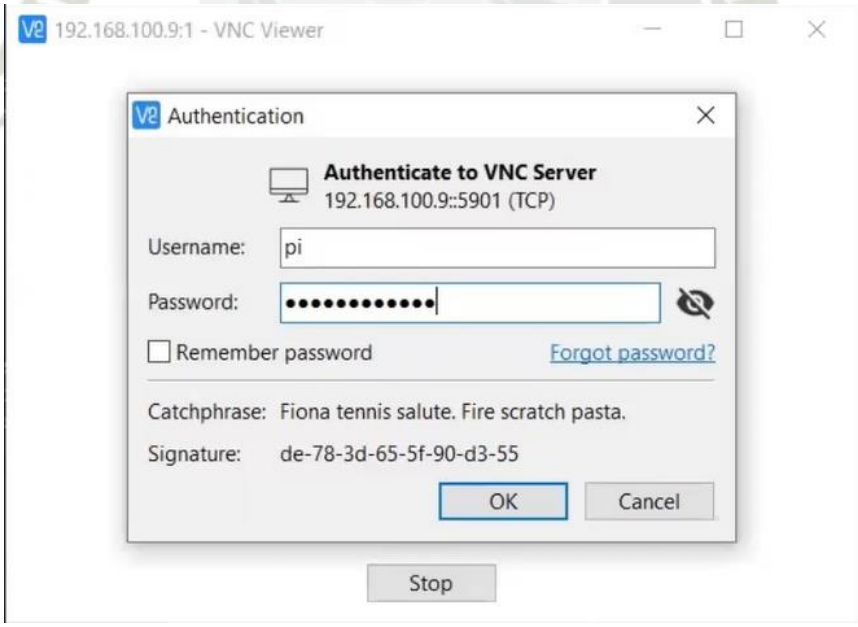


Figura 86. *Visualización de VNC Viewer*

Fuente: Elaboración Propia.

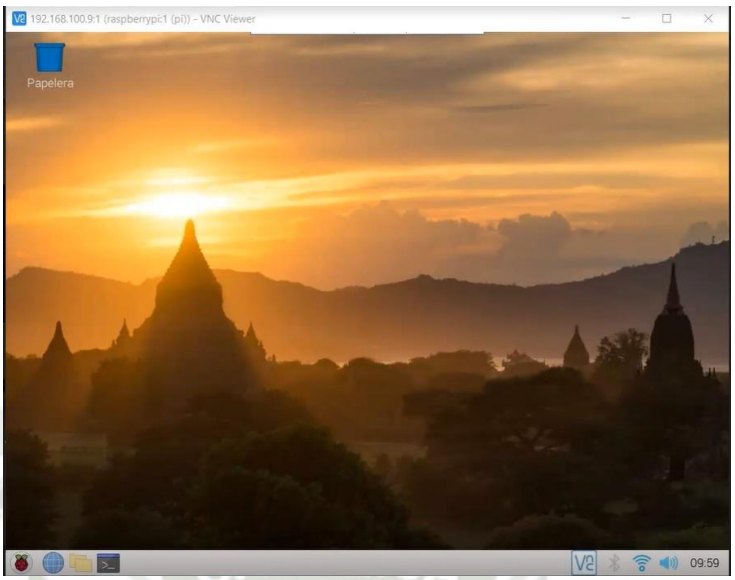


Figura 87. *Vista del escritorio de la tarjeta (Raspberry Pi 4).*

Fuente: Elaboración Propia.

Posteriormente estando dentro correríamos la interfaz de usuario y es la que estaremos visualizando.

4.10. Prueba de la programación de Robot móvil

Antes de realizar la prueba de la programación a través del programa “Python” procedemos a verificar que la bocina bluetooth este activada como vemos en la “Figura 88”, siendo la salida del audio para la bocina.

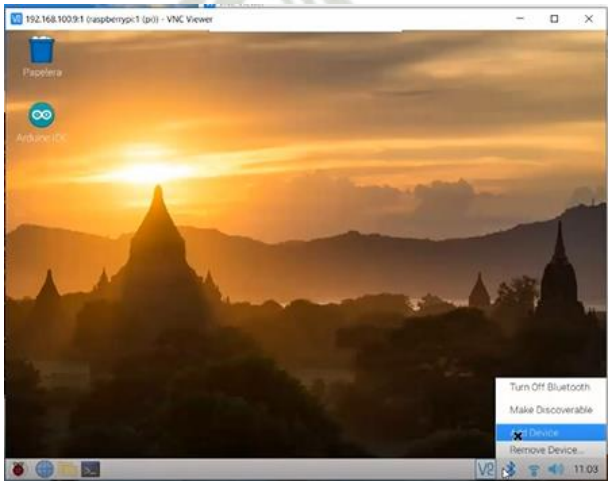


Figura 88. *Pantalla para la verificación de bocina bluetooth.*

Fuente: Elaboración Propia.

Una vez habiendo verificado, proseguimos con el programa “Python”, que se tiene en la interfaz, como se muestra en la “Figura 89”, este programa requiere de bibliotecas, pueden ser propias o que se encuentra disponible en los repositorios. Se hace uso de “pandas o PyGame” (para reproducir el audio).

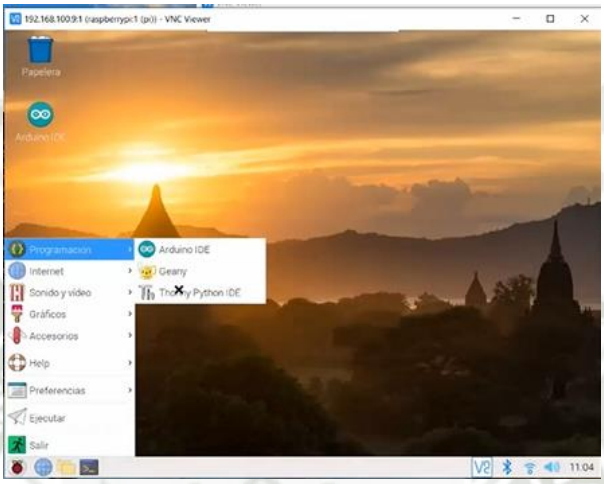


Figura 89. *Ingreso al programa Python.*

Fuente: Elaboración Propia.

main_v1.py - D:\FRANK\TESIS 2021\codigos\main_v1.py (3.9.6)

File Edit Format Run Options Window Help

```
from tkinter import *
from PIL import Image, ImageTk
import sensorMeasure as sm
import RPi.GPIO as GPIO
import numpy as np
import pandas as pd
import pygame, time, cv2, imutils
import seccionPreguntas
import arduinoComunicacion


def iniciar():
    """ Función para iniciar captura de vídeo"""
    global cap
    cap = cv2.VideoCapture(0)
    visualizar()

def visualizar():
    """ Función para mostrar vídeo en la interfaz de usuario """
    global cap
    # revisar si la variable cap contiene información
    if cap is not None:
        # obtener la imagen de la camara
        ret, frame = cap.read()
        if ret == True:
            # redimencionar imagen a tamaño VGA
            frame = imutils.resize(frame, width=640)
            # convertir a espacio de color RGB
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            # convertir imagen tipo Array a tipo Image
            im = Image.fromarray(frame)
            img = ImageTk.PhotoImage(image=im)
```

Figura 90. *Interfaz de bibliotecas de Python.*

Fuente: Elaboración Propia.

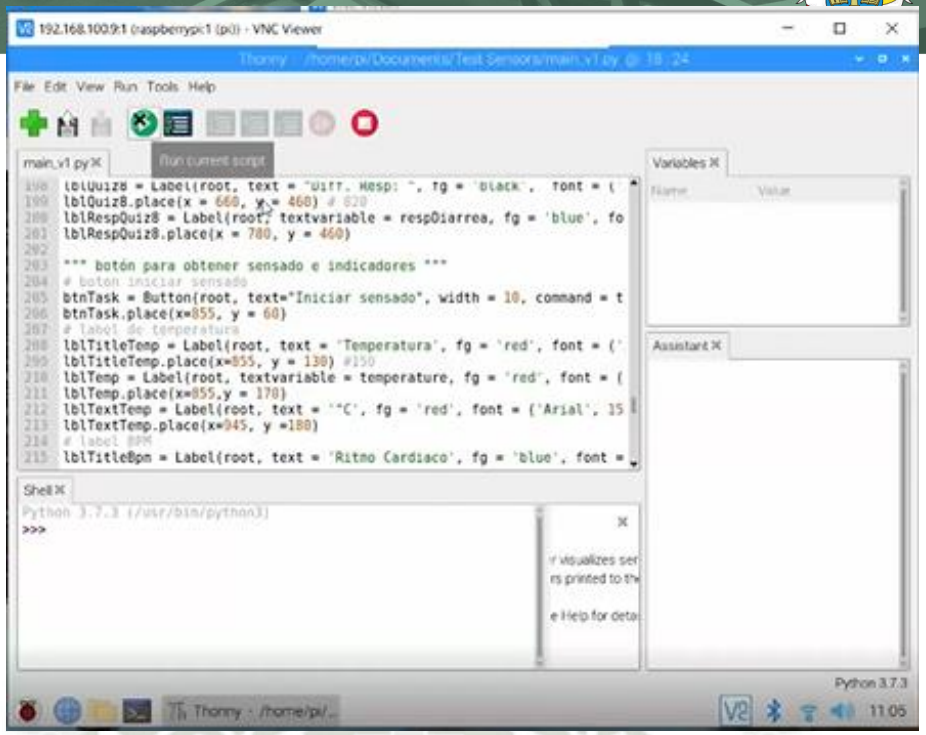


Figura 91. *Códigos que forman parte de la interfaz*.

Fuente: Elaboración Propia.

4.10.1. Prueba de vídeo

Tenemos la interfaz ya completada con los botones superiores de “Iniciar vídeo, finalizar vídeo y cerrar”; en el lado derecho “Iniciar cuestionario e iniciar Sensado” y finalmente en la parte inferior de control manual “izquierda, derecha, adelante y parar” junto al control “automático”, siendo el último el botón “Continuar” como observamos en la figura.

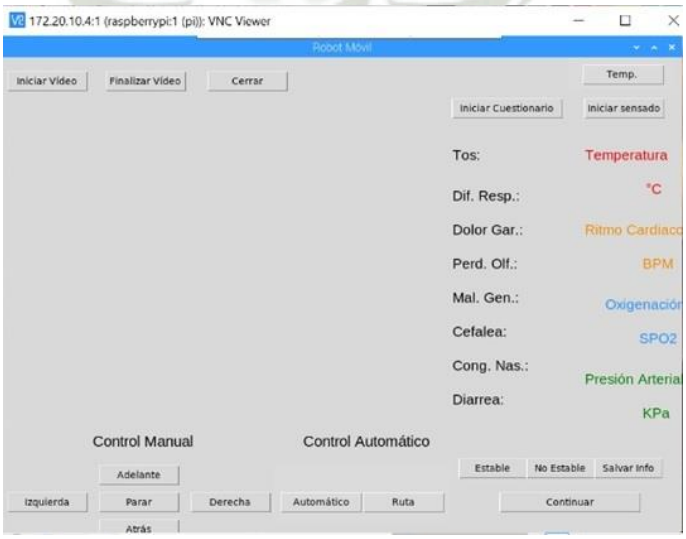


Figura 92. *Interfaz de Robot Móvil.*

Fuente: Elaboración Propia

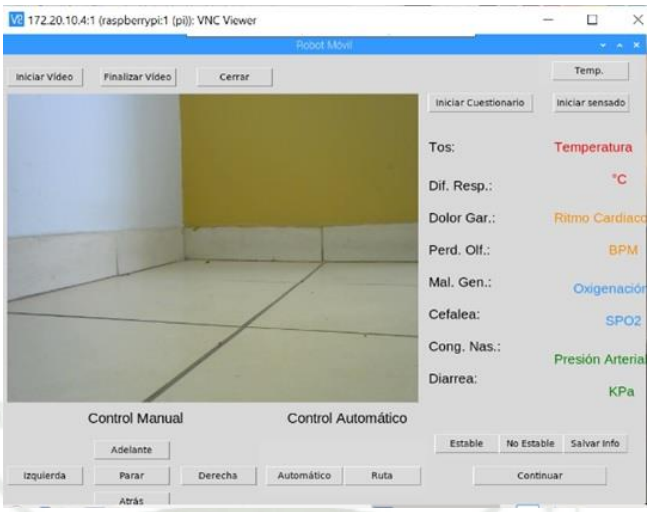


Figura 93. *Inicio del video.*

Fuente: Elaboración Propia.

4.10.2. Prueba del Cuestionario

Al dar inicio al botón de “Iniciar Cuestionario”, nos aparece en la parte inferior las preguntas como observamos en la “Figura 94”, de igual manera, el robot indicará por audio que realizará unas preguntas y como deberá contestarlas acorde a los botones, finalmente el usuario responderá y de acuerdo a eso se mostrará los resultados en la pantalla como vemos en la “Figura 95”.

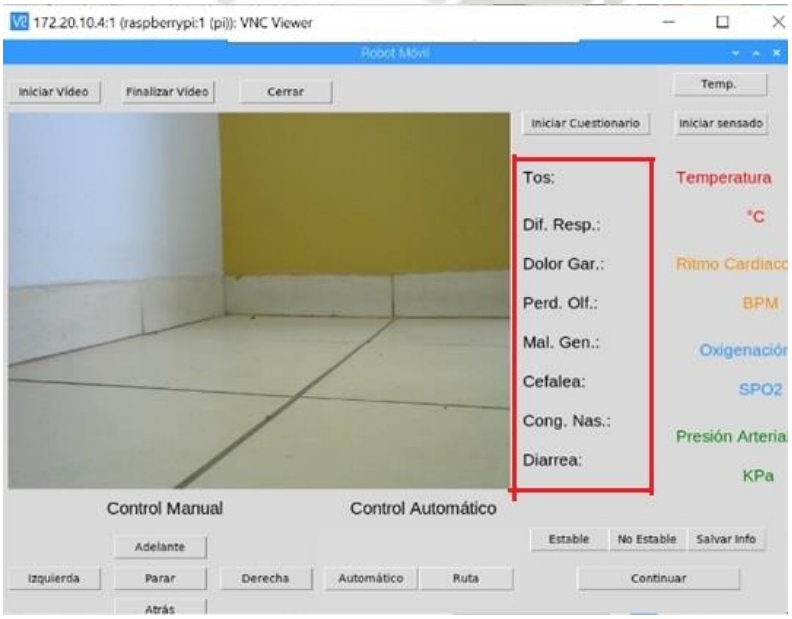


Figura 94. *Visualización de cuestionario.*

Fuente: Elaboración Propia.

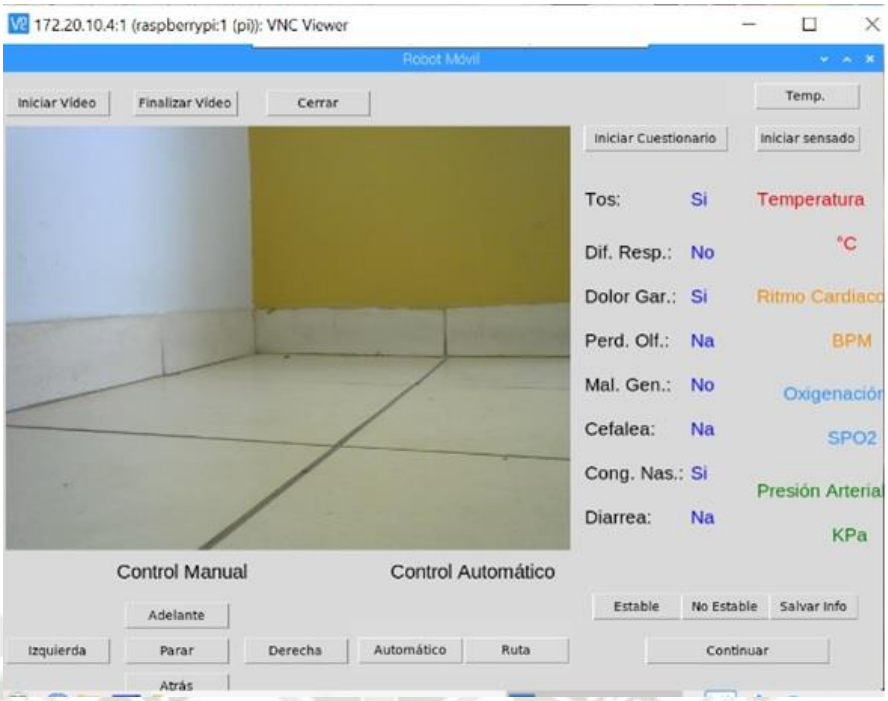


Figura 95. *Resultado de cuestionario*.

Fuente: Elaboración Propia.

4.10.3. Prueba de Sensado

Cuando damos inicio en el botón “Iniciar Sensado” como vemos en la “Figura 96”, el robot nos menciona por vía audio que procederá a realizar la toma de nuestros signos vitales para los cuales nos dará indicaciones para:

- La temperatura(°C): Deberá acercar la mano al sensor.
- El ritmo cardíaco (BPM) y saturación de oxígeno (SPO2): Deberá colocar el dedo índice sobre el sensor del ritmo cardíaco.
- La presión Arterial (KPa): Deberá realizar un total de 5 contracciones.

A la espera de unos minutos, nos dará el resultado de los valores de cada uno de los sensores, dichos valores se dan de acuerdo con la media de varias medidas tomadas.

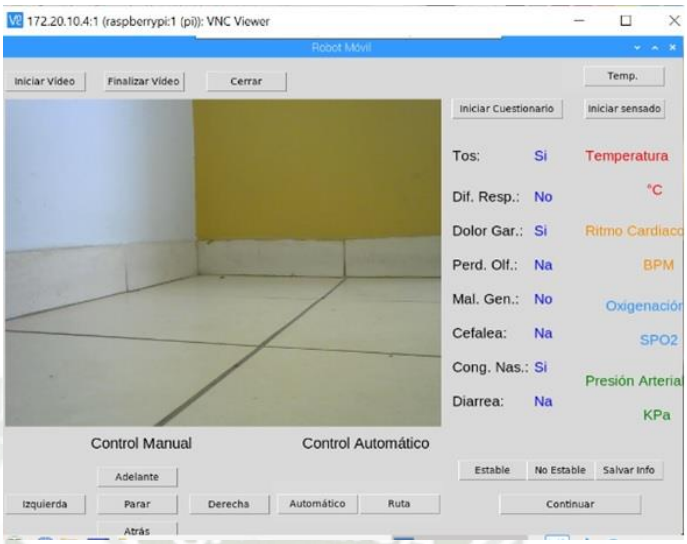


Figura 96. *Visualización del sensado.*

Fuente: Elaboración Propia.

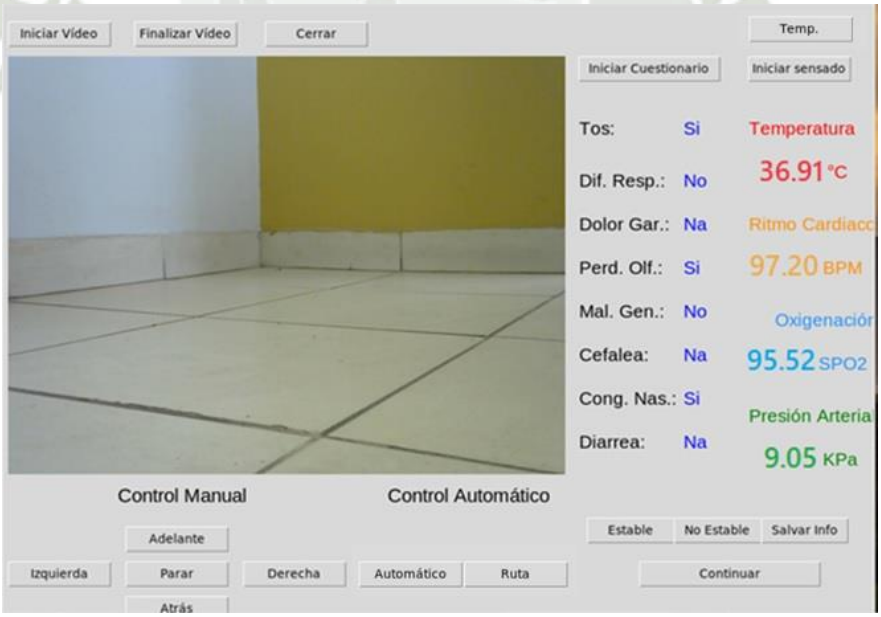


Figura 97. *Resultado de valores del sensado iniciado.*

Fuente: Elaboración Propia.

4.10.4. Finalización de prueba

Tras haber obtenido los valores y estando conforme, procedemos a la finalización de vídeo como vemos en la “Figura 98” y lo cerramos como se muestra en la “Figura 99”.

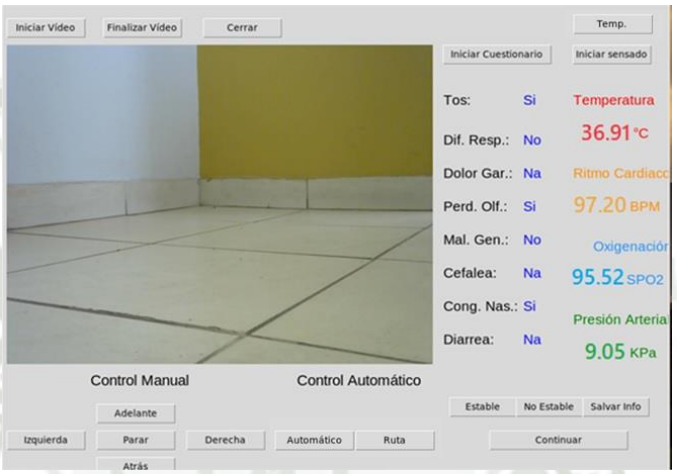


Figura 98. *Visualización de la finalización del video.*

Fuente: Elaboración Propia.

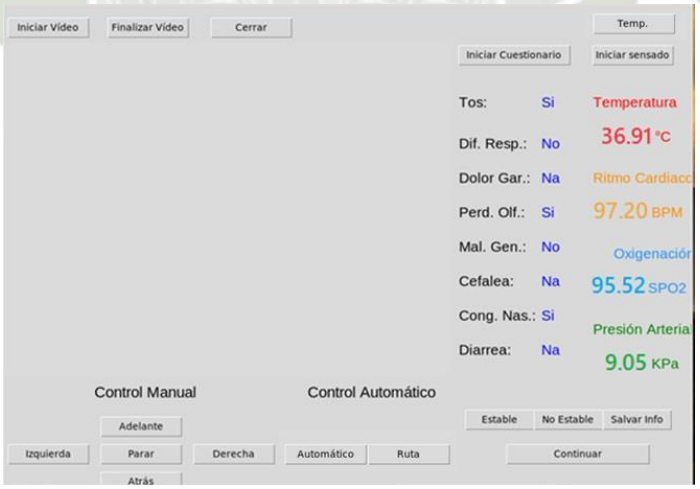


Figura 99. *Visualización del cierre del mismo.*

Fuente: Elaboración Propia.

4.10.5. Comunicación con tarjeta ATMega328

Para el control de los motores DC, que contienen las llantas encargadas de proporcionar movilidad al robot móvil, se utilizó la tarjeta ATMega328. Para esto, se desarrolló un algoritmo, el cual fue codificado en lenguaje C y posteriormente cargado a la tarjeta ATMega328, para que, mediante la comunicación en Serie con la tarjeta Raspberry Pi 4, pudiera ser posible manipular el control de los motores, a través de comandos recibidos por este protocolo de comunicación. Además, se agregó un segundo algoritmo, para llevar a cabo el seguimiento de línea negra, mismo que permite el control automático del robot móvil.

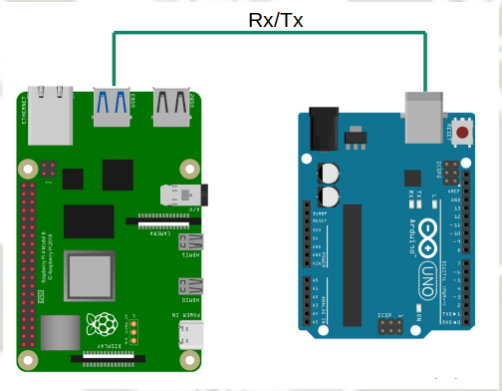


Figura 100. *Representación de comunicación serial, entre las tarjetas Raspberry Pi 4 y ATMega328.*
Fuente: Elaboración Propia.

En la “Figura 101”, se muestra el código principal en lenguaje C de nombre: “RpiComunicacion.ino”, mismo que, demanda el uso de las bibliotecas locales:

- *serialComunication.h*: Biblioteca que lleva a cabo la comunicación Serial con la tarjeta Raspberry Pi 4.
- *sonar.h*: Biblioteca que permite obtener las lecturas de distancia de los sensores HCSR-04, para identificar obstáculos cerca del robot.

Cabe mencionar que, estas bibliotecas son de elaboración propia.

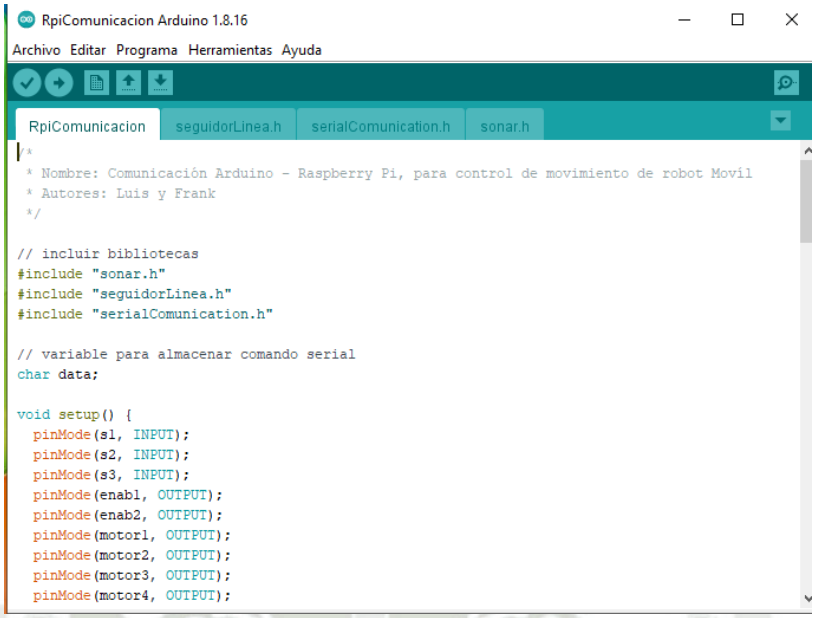


Figura 101.*Captura de código en lenguaje C para la tarjeta ATMega328.*

Fuente: Elaboración Propia

4.11. Apagado de la tarjeta (Raspberry Pi 4)

Tras haber concluido con nuestra prueba procedemos a apagarla debidamente como observaremos en las imágenes siguientes.

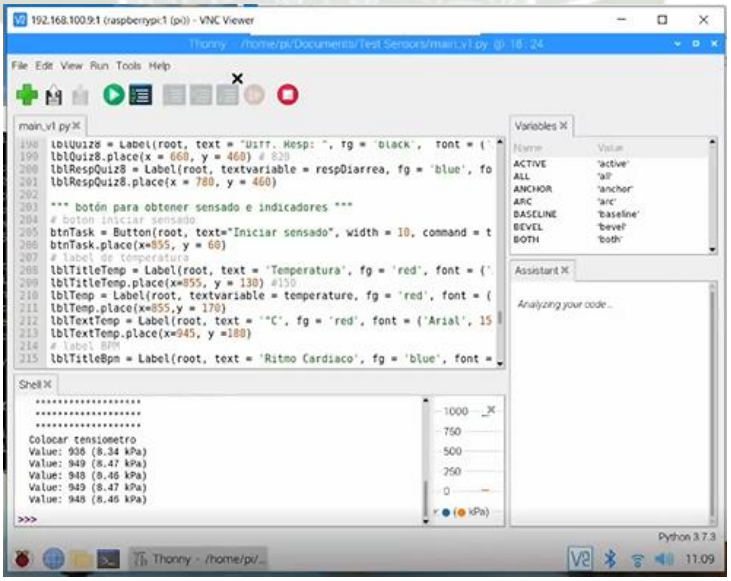


Figura 102. *Cierre de todas las ventanas.*

Fuente: Elaboración Propia.

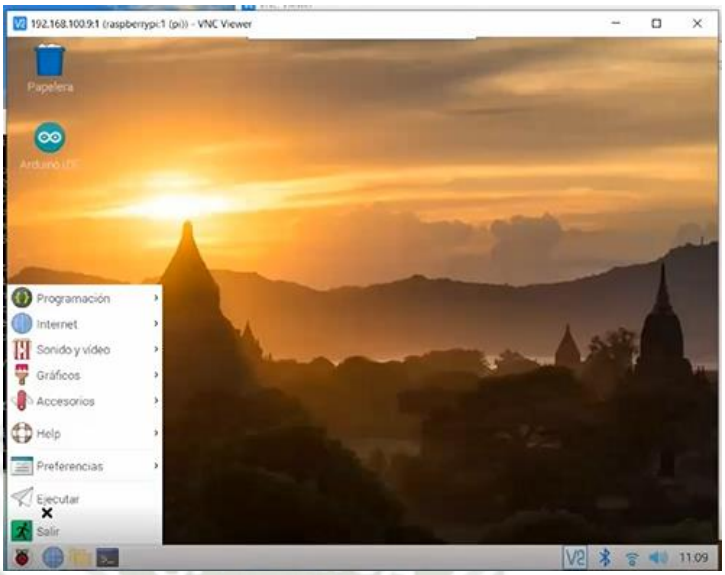


Figura 103. *Visualización del menú de barra de inicio, damos clic en "Salir".*

Fuente: Elaboración Propia.

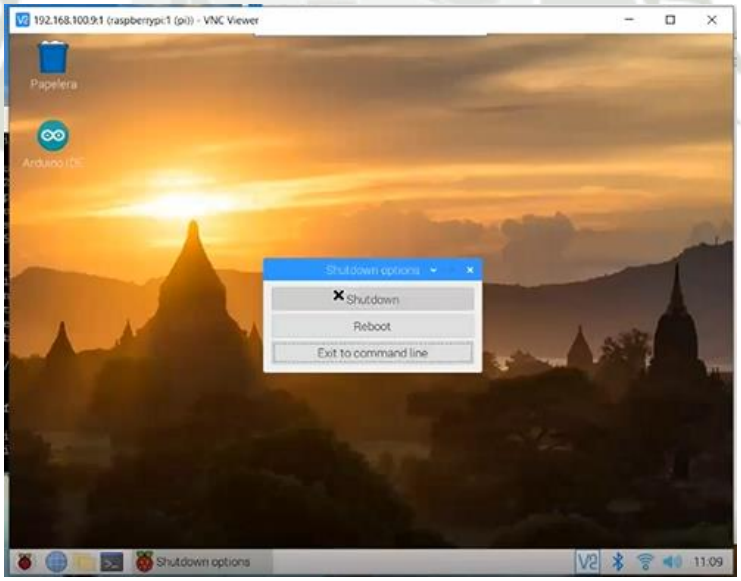


Figura 104. *Pestaña de opciones.*

Fuente: Elaboración Propia.

Nos indica la primera "Apagar", la segunda "Reiniciar" y la tercera "Salir a la línea de comando", damos clic a la primera.

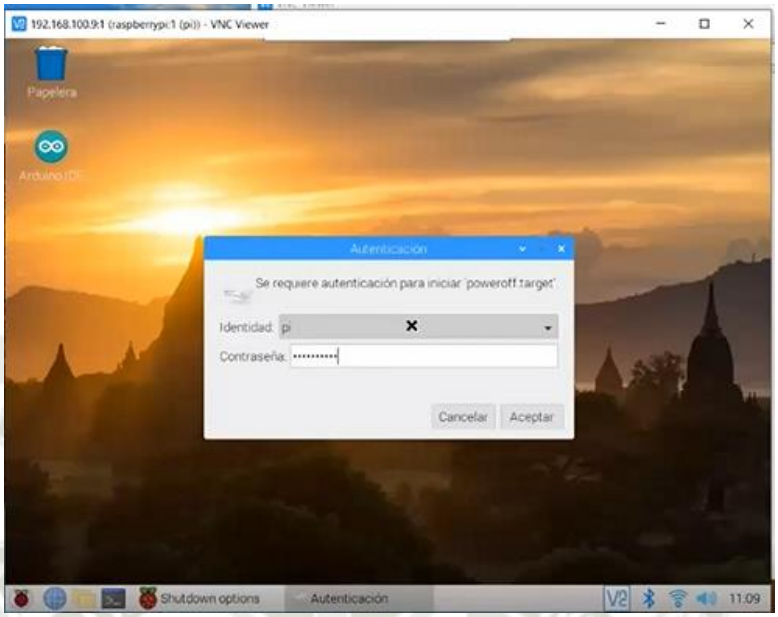


Figura 105. *Pestaña de autentificación.*

Fuente: Elaboración Propia.

Nos muestra una pestaña de autentificación, en la cual colocaremos nuevamente la contraseña de la tarjeta (Raspberry Pi 4).

A continuación, luego de haber hecho las mediciones respectivas de todos los componentes electrónicos de nuestro proyecto, se presentan los pasos empleados para el diseño y construcción del chasis. Finalmente se ensambla todo y se pone en marcha el robot móvil.

Diseñamos las partes que componen el chasis de robot, posteriormente se muestra la fabricación de estas, continuándose con la construcción del prototipo mediante el ensamblaje de las piezas que los componen. A continuación, en la “Figura 106” se muestra las piezas principales necesarias para conformar el chasis del robot, como también en la “Tabla 16” indicándose la función principal a desempeñar, por cada pieza que componen el chasis del robot móvil.

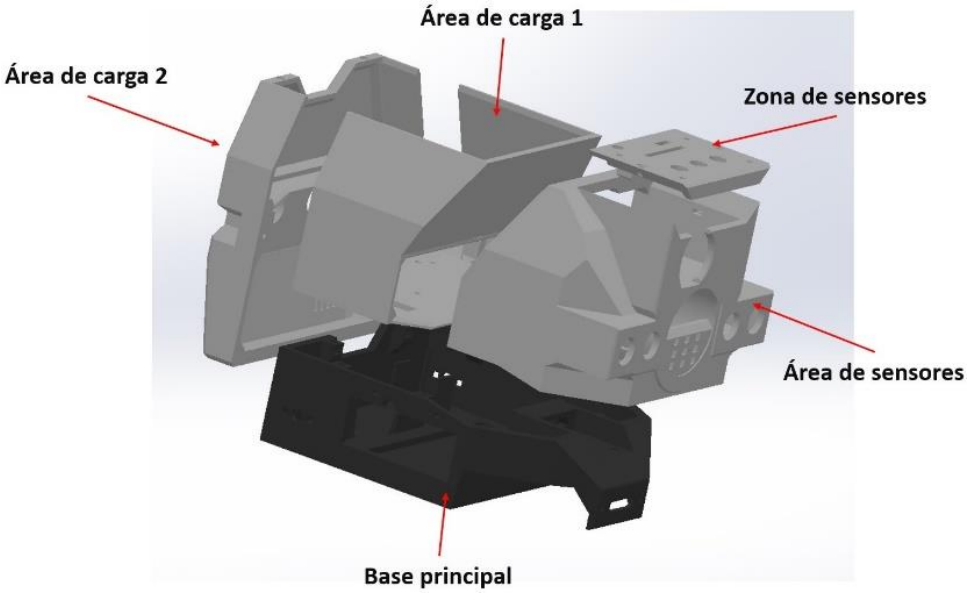


Figura 106. *Representación del prototipo robot móvil.*

Fuente: Elaboración Propia.

Tabla 16.

Descripción de funcionalidad para cada pieza que compone el chasis del prototipo.

Pieza	Función principal
Base principal	Soporte para batería, motores y rueda loca.
Área carga 1	Espacio para material médico y tarjeta de control.
Área carga 2	Soporte para HCSR04 y cierre de espacio para material médico.
Área de sensores	Soporte para HCSR04, bocina, cámara web, bomba de aire.
Zona de sensores	Soporte para sensores y pulsadores.

Fuente. Elaboración Propia.

4.12. Diseño del chasis

Los diseños fueron generados en el software de diseño asistido a computadora (CAD) SolidWorks, versión 2018.

SolidWorks es un software de diseño CAD 3D (diseño asistido por computadora) para modelar piezas y ensamblajes en 3D y planos en 2D. El software que ofrece un abanico de soluciones para cubrir los aspectos implicados en el proceso de desarrollo del producto. Sus productos ofrecen la posibilidad de crear, diseñar, simular, fabricar, publicar y gestionar los datos del proceso de diseño. (Solidbi, 2021, pág. 1)

SolidWorks ofrece soluciones intuitivas para cada fase de diseño. Cuenta con un completo conjunto de herramientas que le ayudan a ser más eficaz y productivo en el desarrollo de sus productos en todos los pasos del proceso de diseño. La sencillez que es parte de su propuesta de valor es decisiva para lograr el éxito de muchos clientes. (Solidbi, 2021, pág. 1)

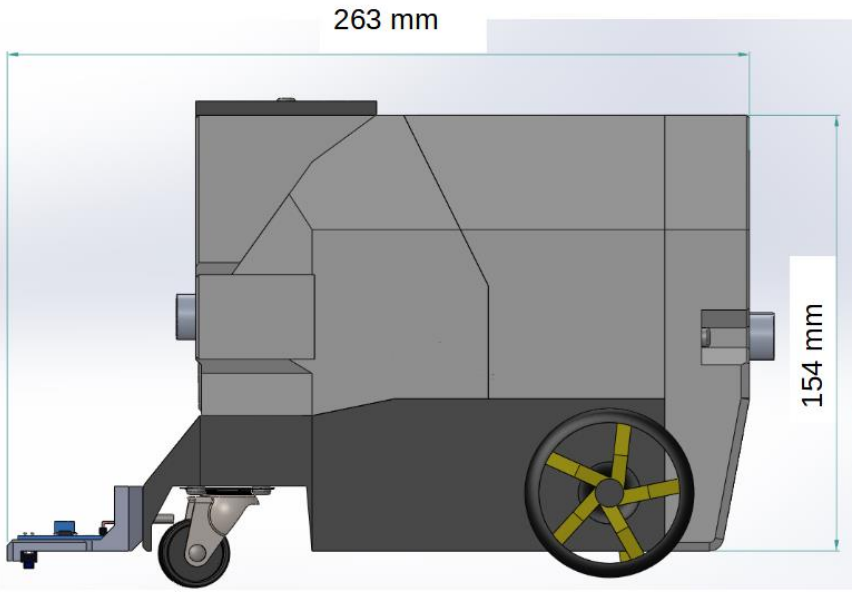


Figura 107. *Medidas laterales del chasis.*

Fuente. Elaboración Propia.

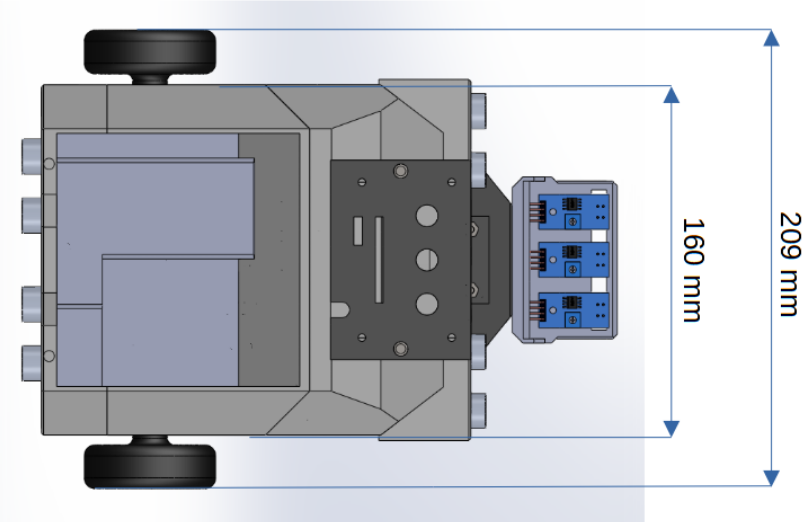


Figura 108.*Medidas inferiores del chasis.*

Fuente: Elaboración Propia.

4.13. Fabricación

Una vez que se diseñaron las piezas, se continuó, con la fabricación de las mismas. El método de fabricación utilizado para esto fue impresión 3D, usando la impresora *Ender 3,* del fabricante *CREALITY*. El tipo de filamento empleado fue PLA, en color negro para la base principal y blanco para las demás piezas. Así mismo, el software que se utilizó para generar los archivos de impresión fue *CURA*. En la “Tabla 17.”, se indican los tiempos de impresión requeridos para la fabricación de cada pieza.

Tabla 17.

Tiempo de impresión por cada pieza que compone el prototipo.

Pieza	Tiempo de fabricación
Base principal	2 días.
Área carga 1	1 día y 18 hrs.
Área carga 2	1 día.
Área de sensores	2 días y 16 hrs.
Zona de sensores	4 hrs.

Fuente: Elaboración Propia.

Figura 109. *Área de sensores.*

Fuente: Elaboración Propia.

Se presenta la pieza del área de sensores, con la colocación de los sensores de distancia (en la parte de los costados), bocina (en la parte central) y cámara web (en la parte superior).

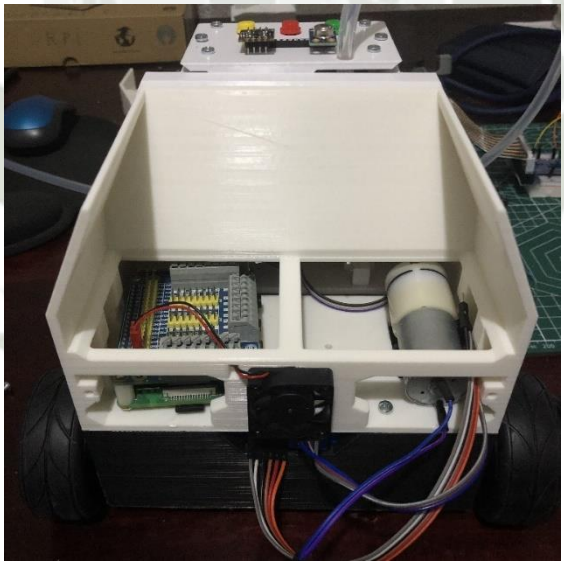


Figura 110. *Vista trasera de ensamble.*

Fuente: Elaboración Propia.

De igual forma, en la Figura anterior se muestra la parte trasera del área de carga 1, la cual contiene en la parte izquierda la tarjeta de control (Raspberry Pi 4), en el centro un ventilador con el fin que la tarjeta de control no se caliente como suele hacerlo y

finalmente en el lado derecho la bomba de aire, sin embargo, ya que realiza mucho ruido en la parte de abajo, ésta se colocó en una pequeña parte del área de carga de la parte superior.

Figura 111. *Vista lateral de ensamble.*

Fuente: Elaboración Propia.

En la Figura anterior se muestra una vista lateral. Se ve también la parte superior donde está el área de carga y el dispositivo para poder medir la presión arterial. Vemos cómo sale la manguera para poder obtener la presión, y se conecta al brazalete de la parte superior. Hay, además, un total de 3 botones para que al paciente pueda responder a algunas preguntas por medio del robot.

Figura 112. *Vista frontal de ensamble.*

Fuente: Elaboración Propia.

Así mismo, en la Figura anterior se muestra la vista frontal del mismo, se puede observar el ensamblado que tiene el robot de los sensores de distancia, la bocina, cámara web y pulsadores.

4.14. Arranque de robot móvil

4.14.1. Sensores empleados para el arranque

Para iniciar con el arranque tuvo que realizarse algunas configuraciones para que los sensores puedan realizar las funciones y realicen una comunicación ininterrumpida.

4.14.1.1. Sensor de presión de aire

El siguiente sensor es un conversor analógico digital, que es empleado debido a que la Tarjeta Raspberry Pi 4 no tiene conversión analógica propia; entonces, se envía la señal de conversión vía comunicación SPI. Para recibir la señal de presión necesitamos el sensor MPX y también a la vez un conversor digital, con comunicación SPI, y ésta información es enviada a la Tarjeta Raspberry Pi 4.

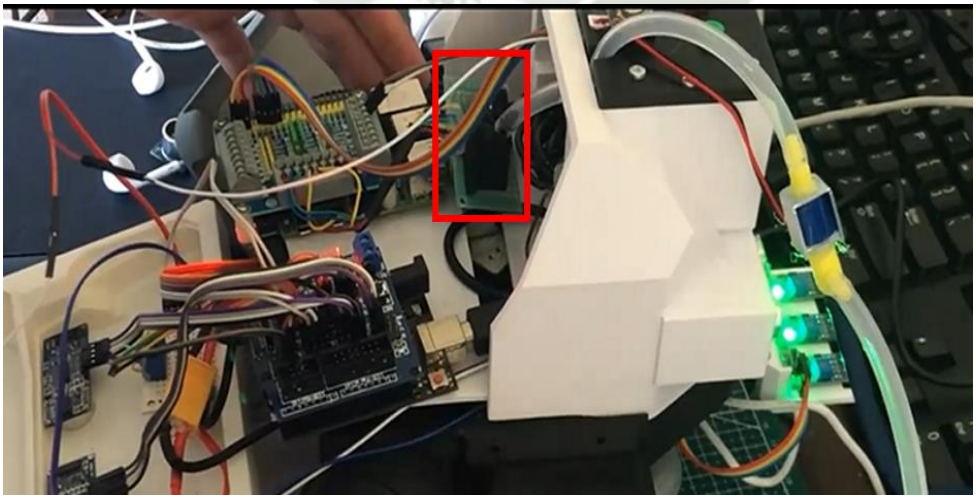


Figura 113. *Sensor de presión de aire, Se tiene un conversor analógico digital.*
Fuente: Elaboración Propia.

4.14.1.2. Sensor de temperatura y ritmo cardíaco

Estos sensores son encontrados en la parte superior negra, así mismo estos sensores se comunican a través del protocolo respectivo.

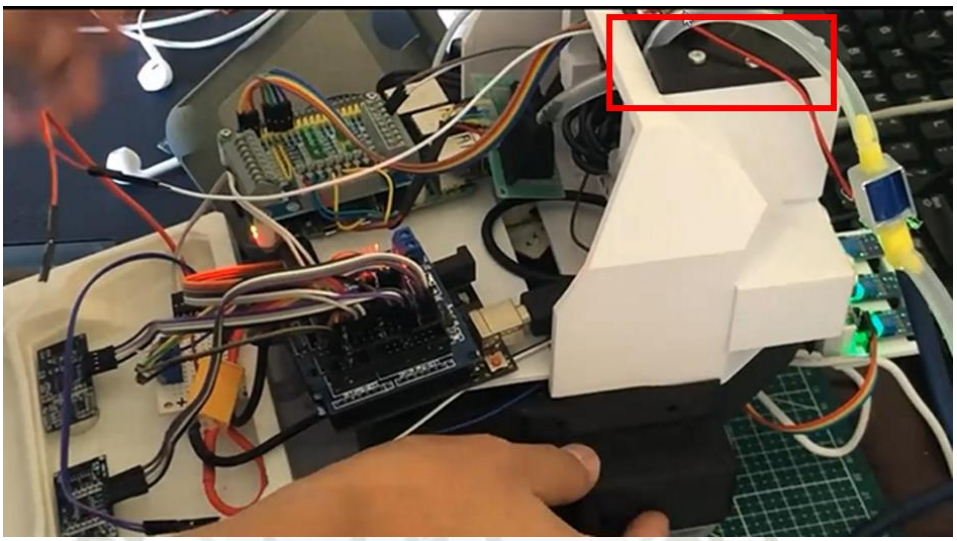


Figura 114. *Sensores de temperatura y ritmo cardiaco.*

Fuente: Elaboración Propia.

Así también, podemos observar cómo están conectados internamente el cable de los sensores de temperatura, ritmo cardíaco y oxigenación, y los botones superiores en la “Figura 115”.

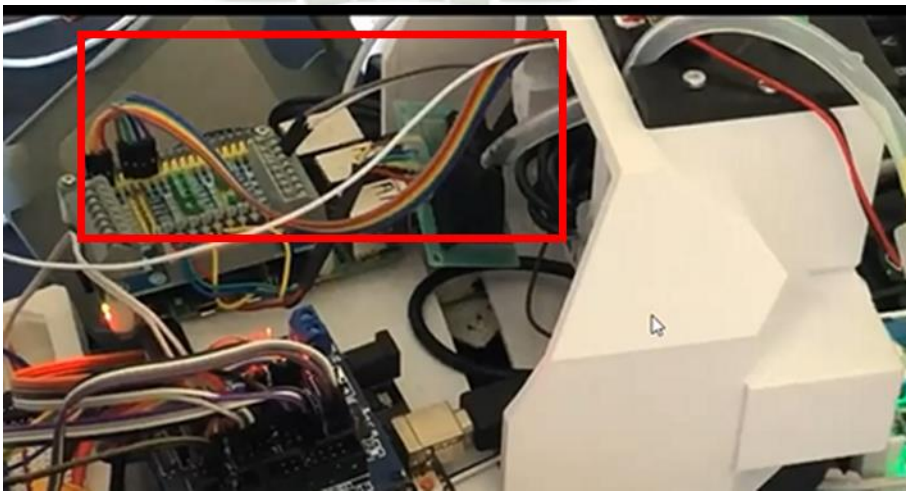


Figura 115. *Conexión de cable de los botones físicos de respuesta.*

Fuente: Elaboración Propia.

4.14.2. Forma de arranque física del robot móvil pre - ensamblado

El robot móvil, como se encuentra en un estado de prueba y por si hubiera algún inconveniente, no fue ensamblado por completo como podremos ver en las siguientes imágenes, así como su alimentación.

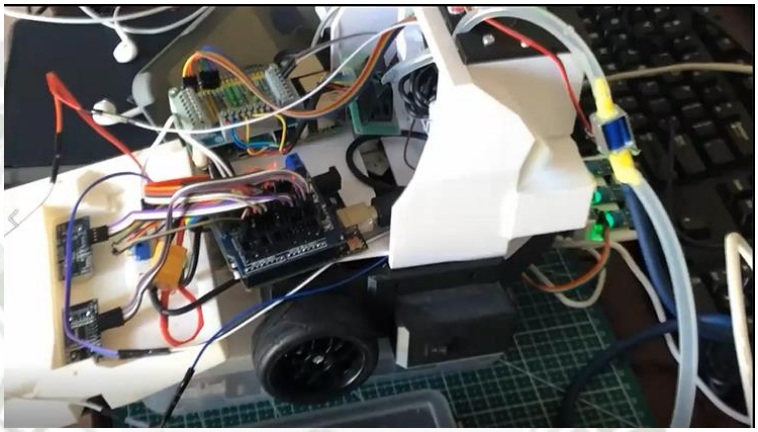


Figura 116. *Visualización exterior del robot móvil.*

Fuente: Elaboración Propia.

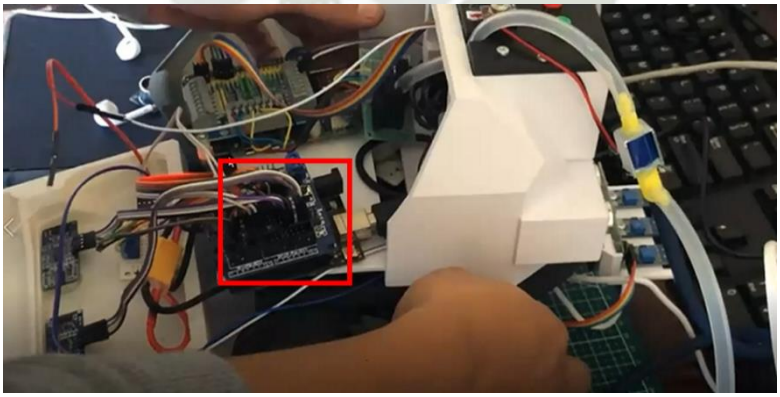


Figura 117.*Tarjeta (Raspberry Pi 4) con su entrada de alimentación que es un puerto USB.*

Fuente: Elaboración Propia.

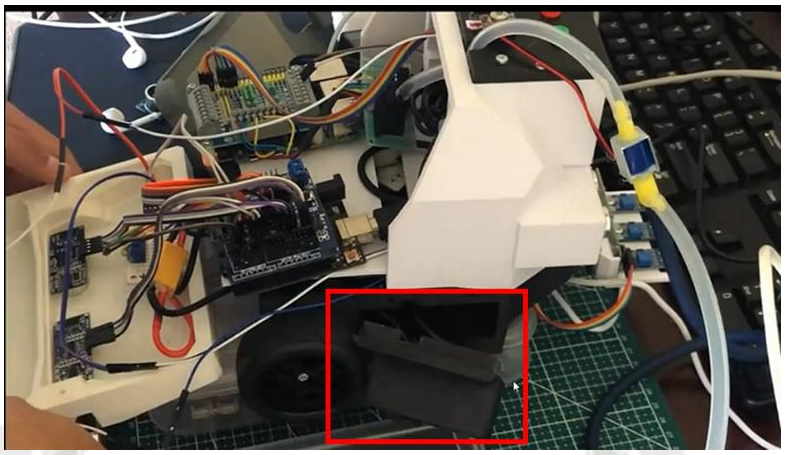


Figura 118. *Ranura de banco de baterías.*

Fuente: Elaboración Propia.

El banco de baterías tiene una entrada USB que se conecta con la Tarjeta (Raspberry Pi 4). Se realizó señalamiento de voltaje y corriente (para alimentar a la tarjeta), con el fin de evitar el sobrecalentamiento.

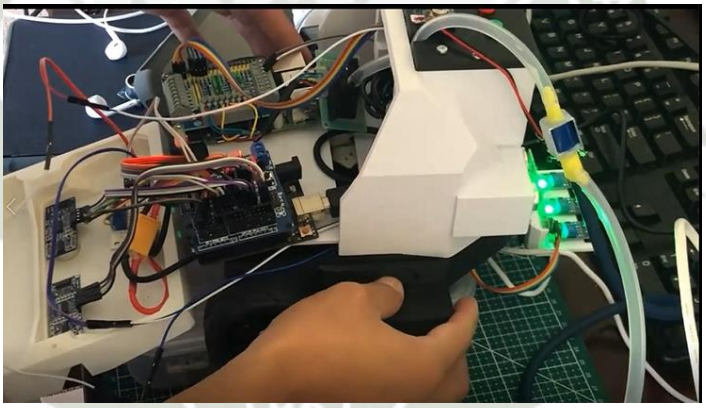


Figura 119. *Ranura de banco de baterías*.

Fuente: Elaboración Propia.

Cerramos la tapa de la “Ranura de banco de baterías “. Una vez que conectamos la entrada USB de la tarjeta, ya podemos notar que está activo al ver las luces del lado derecho.

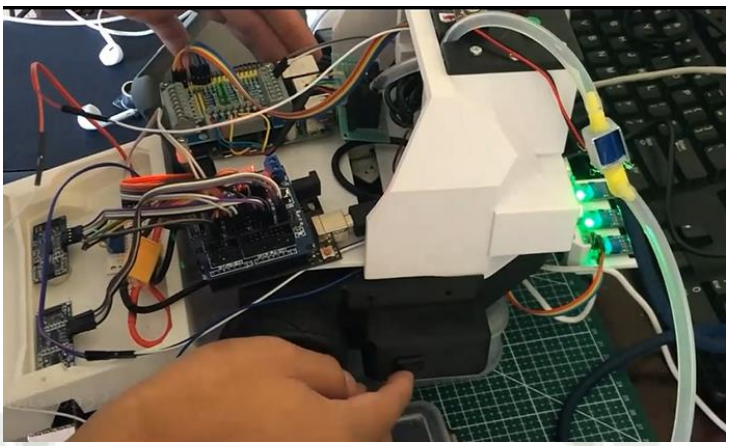


Figura 120. *Switch del botón de la tapa.*

Fuente: Elaboración Propia.

El Switch del botón de la tapa es interruptor de otra batería externa, el cual sirve únicamente para los motores como son las ruedas o la bomba de aire.

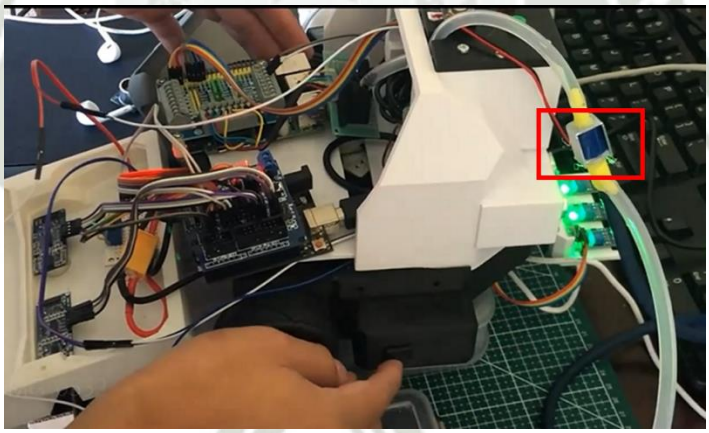


Figura 121. *Electroválvula*

Fuente: Elaboración Propia

La electroválvula es la que nos indica cuándo se realizará el Sensado de la presión que viene del tensiómetro.

Figura 122. *Ranuras para conexión con sensores de temperatura y ritmo cardiaco.*

Fuente: Elaboración Propia.

Así también, tras las previas indicaciones del robot móvil y cuando proceda a realizar las preguntas, el usuario tendrá que responder con los siguientes botones siendo el verde "Si", y el rojo "No".

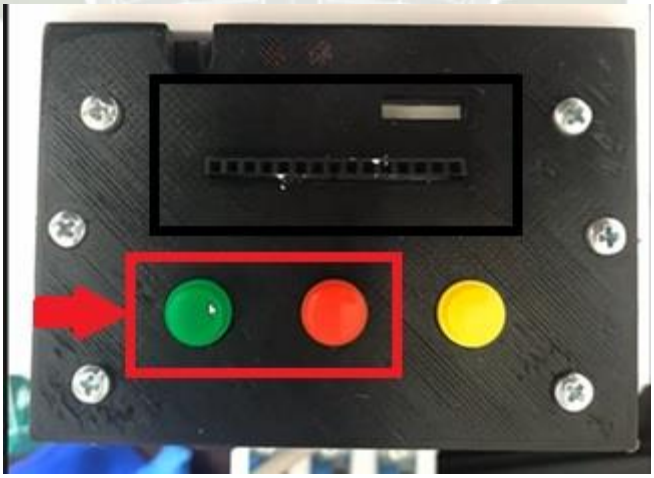


Figura 123. *Visualización de los pulsadores de respuesta.*

Fuente: Elaboración Propia.

Figura 124. *Robot Móvil*

Fuente: Elaboración Propia.

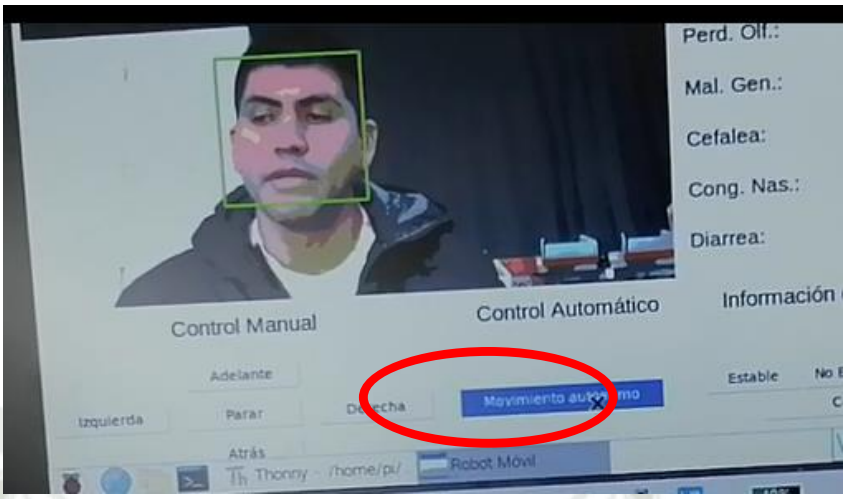
4.15. Demostración en tiempo real

Tomando en cuenta que el robot consta de 2 cámaras para su movimiento automático:

La nueva cámara Raspberry Pi 3/ Pi 4 de 5 MP (cámara 1) es la que lleva a cabo la detección de la ruta hacia las camas de los pacientes, en base al color AZUL y la cámara web HD C505 (cámara 2), nos va a ayudar a recolectar la imagen para hacer reconocimiento de rostros a través del algoritmo Viola - Jones.

Para explicar el funcionamiento, la cámara 1 va a capturar una imagen $\boldsymbol{I_1}$ y la cámara 2 una imagen $\boldsymbol{I_2}$:

1. En primer lugar, se presiona el botón “iniciar video” del interfaz de usuario. Y después, se activa el desplazamiento automático presionando el botón “movimiento automático”.



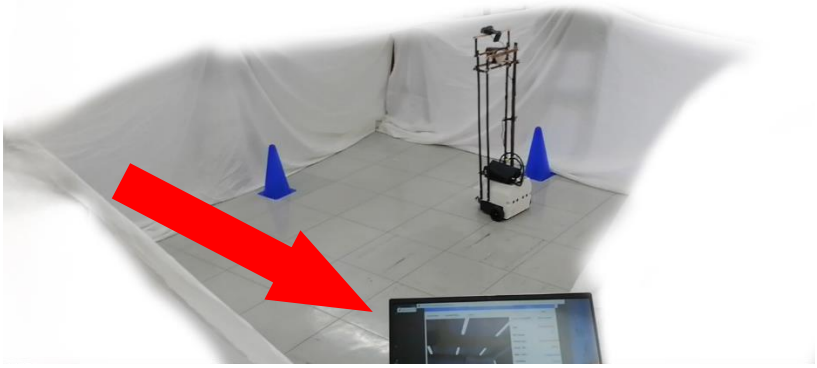
2. Una vez que se presiona el botón “movimiento automático”, se capturan $\boldsymbol{I_1}$ *e* $\boldsymbol{I_2}$.

3. Con $\boldsymbol{I_1}$ se procede a la detección de la ruta hacia las camas, en base a los conos de color AZUL. En este caso sería la detección de los conos, entonces, al detectar un cono se va a ubicar su posición dividiendo la imagen en tres partes: Izquierda, centro y derecha (**L, C, R**). Luego la variable de ubicación tomaría uno de estos valores: **L, C o R**.

4. Se revisará la ubicación detectada. Sea la ubicación en **L** o **R**, el robot se moverá para colocar $\boldsymbol{I_1}$ en **C**, esto lo hacemos a través del envío de comandos al microcontrolador de la tarjeta ATMega328 (mediante comunicación serial).

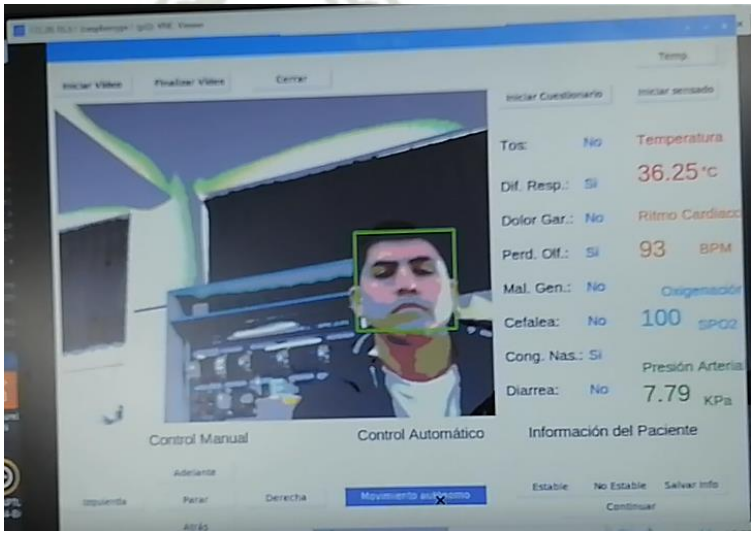
   Entonces, solo si se cumple esto, el robot avanzará hacia adelante hasta detectar el cono a 20 centímetros.

5. Mediante la imagen $\boldsymbol{I_2}$ se verá todo el recorrido del robot en tiempo real a través del interfaz de usuario en la computadora.

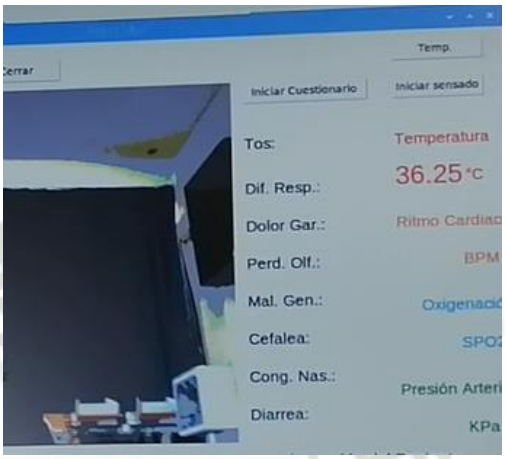


6. Con I_2 procedemos a la detección de rostros para revisar si se ha detectado una cama con “paciente consciente”, “paciente inconsciente” o “sin paciente”.

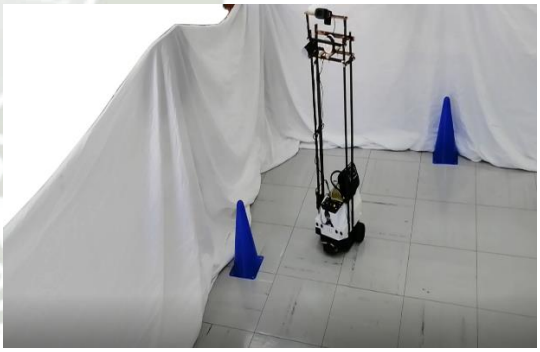
7. Será “paciente consciente” si se ha detectado un rostro. Por tanto, automáticamente el robot comienza a hacer el cuestionario de síntomas COVID, toma los signos vitales y posteriormente muestra todos los datos en el interfaz de usuario de la computadora.



8. Será “paciente inconsciente” si se ha detectado un rostro, pero no responde al cuestionario de síntomas COVID. Por tanto, automáticamente toma solo la temperatura y posteriormente muestra este dato en el interfaz de usuario de la computadora.

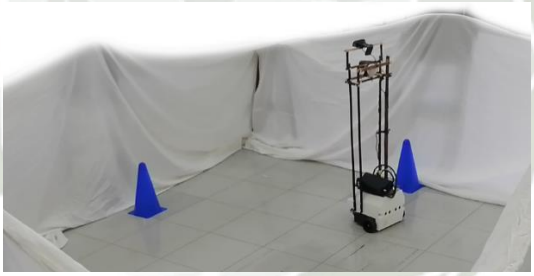


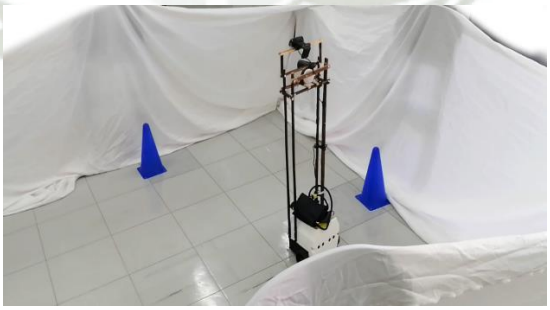
9. La cama estará “sin paciente” si no detecta un rostro. Por tanto, el robot evadirá el cono.



10. Una vez que se termine la evaluación anterior, el robot retrocede y posteriormente gira 45° hacia la izquierda. Si se ha tomado datos, el robot permanece 20 segundos mostrándolos en pantalla hasta que se guardan automáticamente en la carpeta de “pacientes” en la computadora, y luego, continuará su recorrido buscando el siguiente cono con $\boldsymbol{I_1}$. Caso contrario, si no hay paciente, simplemente continúa su recorrido.

11. Durante la búsqueda del siguiente cono para su recorrido: Si no se detecta ninguno luego del giro, el robot seguirá girando a la izquierda hasta encontrarlo, y así, va a repetir los pasos 3 a 10.





4.16. Presupuesto del prototipo de robot

Nombre	Cantidad	Precio USD	Descripción
Raspberry Pi 4	1	150	Tarjeta controladora
ATMega328 Arduino UNO	1	14	Tarjeta controladora
Shield sensor para Raspberry Pi 4	1	10	Conexión de sensores
Shield sensor para ATMega328 Arduino UNO	1	6	Conexión de sensores
Micro SD de 64GB	1	9.25	Memoria para instalar SO de Raspberry Pi
HCSR04	4	5	Sensor de distancia
Módulo AMG8833	1	200	Sensor de temperatura
MAX30100	1	10	Sensor de ritmo cardíaco
MPX10DP	1	32.5	Sensor de presión de aire
Motorreductor Pololu 25D 6-12 VDC 2A 80RPM	2	45	Movimiento del robot
Llanta para motor Pololu 25D	2	37.5	Movimiento del robot
Bomba de aire 3-6 V 0.37-0.62A 8 - 13 PSI	1	12.5	Generación de presión de aire
Manguera de plástico 50cm	1	7.5	Para transporte de aire
Electroválvula 6v N/C	1	10	Para control de circulación de aire
Cámara Web Logitech	1	45	Para reconocimiento facial
Cámara Raspberry Pi 3/Pi4 5 MP	1	22	Para reconocimiento de camino de conos azules
Bocina bluetooth Xiaomi	1	30	Reproducción de sonido
Pulsadores	3	1	Recibir respuesta de usuario
Regleta de pines hembra	1	1	Conexión de sensores
Regleta de pines macho	1	1	Conexión de sensores
Powerbank	1	42.5	Alimentación de tarjetas de control y sensores
Batería LiPo 7VDC 1 Amp	1	37.5	Alimentación de motores
Clema de 3 terminales	3	1	Conexión de alimentación
Interruptor	1	1.5	Encendido/apagado de robot
Conector XT60 macho	1	2.5	Conector de alimentación
Cable flexible rojo 30 cm	1	2.5	Conexión de alimentación
Cable flexible negro 30 cm	1	2.5	Conexión de alimentación
Cables dupont H-H	2	7.5	Conexión de sensores
Cables dupont M-H	1	4.25	Conexión de sensores

Placa perforada	4	4	Circuito de sensores para seguidor de línea
Ventilador 5v	1	4.25	Ventilación de tarjeta de control
Brazalete tamaño mediano	1	15	Para toma de presión de aire
Módulo para ULN2003	1	2.5	Control de bomba de aire
Integrado MCP3008	1	10	Convertidor de analógico a digital
Módulo L298D	1	4.75	Para control de motor DC
Cable USB tipo A a tipo B	1	4.25	Para conectar Arduino con Raspberry Pi
Cable USB tipo A a tipo C	1	4.5	Para conectar Raspberry Pi con Power bank
Tornillería	50	7.5	Tornillos y tuercas tamaño M3 para ensamble del robot móvil
Rueda loca	1	4.25	Para movimiento del robot
Rollo de filamento PLA color Negro	0.5kg	21	Para fabricación del chasis del robot
Rollo de filamento PLA color blanco	0.5kg	21	Para fabricación del chasis del robot
Rollo de filamento PLA color amarillo	0.5kg	21	Para fabricación del chasis del robot
	Total	**875.00**	**S/. 3,325**

Tabla 18. *Presupuesto detallado*
Fuente: Elaboración propia

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

1) Se logró diseñar e implementar el prototipo de un robot móvil con la tarjeta principal Raspberry Pi 4. Que reemplace las labores de un enfermero asistente en tiempos de pandemia Covid-19, que aún padece la población de la ciudad de Arequipa.
2) Se pudo elaborar el mecanismo y la programación para el desplazamiento del robot, activado remotamente y que siga automáticamente una guía azul hasta encontrar al paciente (mediante reconocimiento facial). Para esto se contó también con el apoyo del microcontrolador ATMega328, que se encarga del movimiento de los motores, y así, aumentar la eficiencia del prototipo.
3) Se diseñó, ensambló y programó el mecanismo de monitoreo de pacientes con Covid-19, gracias a la selección exitosa de sensores de presión arterial, oxigenación, ritmo cardíaco y temperatura, para obtener estos signos vitales de los pacientes.
4) La elaboración de un sistema para la comunicación a distancia (incluyendo cuestionario de síntomas COVID), entre el paciente y el personal de salud en la PC principal, resultó ser adecuada para evitar el contagio por contacto directo entre ambas partes.
5) Se elaboró una conexión WIFI exitosa para establecer la comunicación inalámbrica entre la PC principal y el robot móvil.
6) Se logró diseñar y programar una aplicación en la PC, que muestre y entregue al personal de salud, un registro de los signos vitales tomados del monitoreo al paciente.
7) Se logró construir la estructura base del robot prototipo, incluyendo un espacio para que éste lleve material médico con un peso máximo de 1kg.

5.2. RECOMENDACIONES

- Se recomienda hacer una investigación tecnológica más profunda sobre cómo se está apoyando, en otros países, durante casos emergentes de enfermedades pandémicas. Con el fin de estudiar la información, y ver de qué forma se puede mejorar e implementar una solución, en nuestra ciudad y en nuestro país.

REFERENCIAS BIBLIOGRÁFICAS

ADSONG. (2020). *Digital relative humidity & temperatura sensor DHT11.* ADSONG.

Aguilera, M., Bautista, M., & Iruegas, J. (2007). Diseño y control de Robots Móviles. *Mecamex*. Recuperado el 01 de 03 de 2021, de http://www.mecamex.net/anterior/cong02/papers/art24.pdf

Agurto, D. (2020). *Integración de un sistema de adquisición de datos mediante el uso de un Arduino Mega y Raspberry Pi3 como servidor web y base de datos.* Obtenido de Universidad Nacional de Piura: https://repositorio.unp.edu.pe/bitstream/handle/20.500.12676/2477/IEYT-AGU-VAL-2020.pdf?sequence=1&isAllowed=y

ALLDATASHEET. (2020). *L298 Datasheet (PDF) - STMicroelectronics.* ALLDATASHEET.

ALLDATASHEET. (2020). *SRF05 Datasheet (PDF) - List of Unclassifed Manufacturers.* ALLDATASHEET.

Bambino, I. (2008). Recuperado el 15 de 02 de 2021, de Una introducción a los Robots Móviles: https://aadeca.org/pdf/CP_monografias/monografia_robot_movil.pdf

Barrientos, V., García, J., & Silva, R. (2007). Robots móviles: Evolución y Estado del Arte. *Polibits*, 12-17. Recuperado el 08 de 02 de 2021, de https://www.redalyc.org/pdf/4026/402640448003.pdf

BricoGeek. (13 de 12 de 2021). *Array de sensores infrarrojos QTR-8RC (Digital)*. Obtenido de Sensor Infrarrojo TCRT5000: • https://naylampmechatronics.com/sensores-proximidad/39-sensor-infrarrojo-tcrt5000.html

Camacho, E., & Escobedo, E. (2019). *Sistema Robótico movil para el transporte de materiales ligeros.* Obtenido de Instituto Politécnico Nacional-Mexico: https://tesis.ipn.mx/jspui/bitstream/123456789/28235/1/tesis_final.pdf

Chitay Bautista, J. A. (2020). *Dispositivo que mide el índice ultravioleta, sobre un paciente para prevenirlo del cáncer de piel, aplicando internet de las cosas.* Guatemala: Universidad de San Carlos de Guatemala.

Cooking-hacks. (s.f.). *e-Health Sensor Shield V2.0 for Arduino, Raspberry Pi and Intel Galileo [Biometric / Medical Applications]* . Obtenido de Cooking-hacks: https://www.cooking-hacks.com/ehealth-sensor-shield-biometric-medical-arduino-raspberry-pi.html

Cornejo, J., Cornejo-Aguilar, J., & Perales-Villarroel, J. (octubre de 2019). *Innovaciones Internacionales en Robótica Médica para Mejorar el Manejo del Paciente en Perú.* Obtenido de Revista Facultad de Medicina Humana URP 19(4). Pp. 105-113: http://www.scielo.org.pe/pdf/rfmh/v19n4/a16v19n4.pdf

Cornejo, J., Vargas, M., & Cornejo-Aguilar, J. (Octubre de 2020). *Aplicaciones innovadoras de la robótica y biomédica en la salud pública durante la pandemia del COVID-19.* Obtenido de Revista Facultad de Medicina Humana URP. 20(4). Pp. 756-757: http://www.scielo.org.pe/pdf/rfmh/v20n4/2308-0531-rfmh-20-04-756.pdf

COVANTEC. (2019). *Programación en Phyton - Nivel básico*. Recuperado el octubre de 2021, de Ventajas y Desventajas: https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/ventajas_desventajas.html

Dexter Industries. (2014). *The "JJ" WiFi Car*. Obtenido de Dexterindustries.com: https://www.dexterindustries.com/BrickPi/brickpi-tutorials-documentation/projects/wifi-car/

Dignani, J. (2011). (Trabajo final integrador de especialización en Redes y Seguridad). *Análisis del Protocolo Zigbee*. Buenos Aires, Argentina: Universidad Nacional de la Plata. Recuperado el 01 de 03 de 2021, de http://sedici.unlp.edu.ar/bitstream/handle/10915/18349/Documento_completo__.pdf?sequence=1&isAllowed=y

ECKSTEIN. (2020). *POLOLU METAL GEARMOTOR 25D 12V WITH 48 CPR ENCODER.* ECKSTEIN KOMPONENTE.

ELEC FREAKS. (2020). *Ultrasonic Ranging Module HC - SR004.* ELEC FREAKS.

Espinosa Muñoz, J. I. (2015). *Paradigma de Programación. Comparación de los lenguajes de Dataflow LabVIWE y VEE.* [Trabajp de Investigación], Universidad Politecnica

Madrid, Madrid. Obtenido de https://es.slideshare.net/javierflip/comparacin-de-los-lenguajes-de-dataflow-labview-y-vee

Flores, L., & Garay, M. (2017). Diseño e implementación de un robot móvil de desplazamiento autónomo basado en un controlador proporcional derivativo. *(Trabajo de Grado para optar el Título Profesional de Ingeniero Electrónico)*. Callao, Perú: Universidad Nacional Del Callao. Recuperado el 03 de 02 de 2021, de https://alicia.concytec.gob.pe/vufind/Record/UNAC_f2596c58e132c21bd18f4e868f740308

Gonzales, S. (2021). *Arduino Ventajas y desventajas*. Recuperado el octubre de 2021, de https://es.scribd.com/document/448301825/ARDUINO-VENTAJAS-DESVENTAJAS-docx

Grayson, D. (23 de octubre de 2013). *PiBot-B: mobile robot with a Raspberry Pi*. Obtenido de Pololu.com: https://www.pololu.com/blog/233/pibot-b-mobile-robot-with-a-raspberry-pi

Kandola, A. (14 de 04 de 2020). *Medical News Today*. Recuperado el 05 de 02 de 2021, de Causas del Coronavirus; Su origen y cómo se propaga: https://www.medicalnewstoday.com/articles/es/causas-del-coronavirus-su-origen-y-como-se-propaga

MaxBotix. (2020). *LV - MaxSonar®-EZ™ Series.* MaxBotix.

Maxim integrated. (2020). *MAX30101.* Obtenido de https://datasheets.maximintegrated.com/en/ds/MAX30101.pdf?__cf_chl_captcha_tk__=pmd_aw.hPcQjZFirjSZlycprDAMbqjGhAa3qaaxGQGT.Vb0-1634678598-0-gqNtZGzNAzujcnBszQi9

Maxim integrated. (2020). *MAX30105 - High-Sensitivity Optical Sensor for Smoke Detection Applications.* Maxim integrated.

Maxim integrated. (2020). *MAX30102 - High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health.* Maxim integrated.

MELEXIS. (2020). *MLX90614 family.* Melexis.

Micro Controladores. (2021). *Todo sobre Microcontroladores*. Recuperado el octubre de 2021, de https://microcontroladoress.com/

Mochales Mielgo, M. (28 de setiembre de 2020). *Python, el lenguaje de programación más popular de 2020*. Obtenido de Profile: https://profile.es/blog/python/

Naylamp mechatronics. (s.f.). *PULSIOXÍMETRO MAX30102*. Obtenido de Naylamp mechatronic: https://naylampmechatronics.com/biomedico/444-pulsioximetro-max30102.html

Naylamp Mechatronics. (s.f.). *SENSOR ULTRASONIDO HC-SR04*. Obtenido de Naylamp Mechatronics: https://naylampmechatronics.com/sensores-proximidad/10-sensor-ultrasonido-hc-sr04.html

NXP. (octubre de 2008). *Freescale Semiconductor*. Recuperado el octubre de 2021, de https://www.nxp.com/docs/en/data-sheet/MPX2050.pdf

NXP. (2021). *Especificaciones técnicas de MPX2050D*. Recuperado el octubre de 2021, de https://www.nxp.com/part/MPX2050D#/

OAS;. (s.f.). *Metodología Top-Down*. Recuperado el 19 de octubre de 2021, de http://163.10.22.82/OAS/modularizacion/metodologa_topdown.html

Omega. (13 de 12 de 2021). *Omega a espectris company.* Obtenido de https://es.omega.com/prodinfo/transductores-de-presion.html

OpenWebinars. (22 de julio de 2019). *Qué es C++ : Características y aplicaciones.* Obtenido de https://openwebinars.net/blog/que-es-cpp/

OSI. (21 de 10 de 2021). *OSI. Oficina de seguridad del Internauta.* Obtenido de https://www.osi.es/es/herramientas-gratuitas/vnc-viewer

PANASONIC. (2020). *Infrared Array Sensor Grid-EYE (AMG88).* Panasonic.

Pérez, H. (2019). (Tesis para obtener el Título de Ingeniería en Computación). *Robot móvil con percepción auditiva y sistema de control a distancia*. Atlacomulco, México: Universidad Autónoma del Estado de México. Recuperado el 01 de 02 de 2021, de http://ri.uaemex.mx/bitstream/handle/20.500.11799/104957/Tesis_Heivilina-Perez-

Arias%20sin%20oficios.pdf;jsessionid=CFBBBF9020AFC95ECE209F21765DD55D?sequence=3

Pez Nuss. (29 de enero de 2020). *Conociendo Raspberry Pi*. Recuperado el octubre de 2021, de https://peznuss.medium.com/conociendo-raspberry-pi-3dd077b12c92

POLOLU. (2015). *QTR-8A and QTR-8RC Reflectance Sensor Array User's Guide.* POLOLU.

POLOLU. (2016). *QTR1A Reflectance Sensor (2Pack).* POLOLU.

Naylamp Mechatronics. (s.f.). Obtenido de Pulsioxímetro MAX30102: https://naylampmechatronics.com/biomedico/444-pulsioximetro-max30102.html

Ramírez, R., & Reyes, R. (2015). (Tesis para optar el Título de Ingeniero Electrónico). *Diseño e Implementación de un robot autónomo móvil usando tecnología FPGA*. Guayaquil, Ecuador: Universidad Politécnica Salesiana Ecuador. Recuperado el 01 de 02 de 2021, de https://dspace.ups.edu.ec/bitstream/123456789/10429/1/UPS-GT001506.pdf

Robokits India. (s.f.). *INFRARED TEMPERATURE SENSOR GY-906 MLX90614*. Obtenido de Robokits India: https://robokits.co.in/sensors/temperature-humidity/infrared-temperature-sensor-gy-906-mlx90614

ROBOT - HEAD TO TOE. (2013). *CYTRON TECHNOLOGIES.* ROBOT - HEAD TO TOE.

Salcedo, R. (2018). (Trabajo de Grado para optar el Título Profesional de Ingeniero Electrónico). *Desarrollo de un discriminador de calidad de un sistema de control y monitoreo embebido basado en tecnología FPGA*. Arequipa, Perú: Universidad Nacional de San Agustín de Arequipa. Recuperado el 03 de 02 de 2021, de http://repositorio.unsa.edu.pe/bitstream/handle/UNSA/8266/IEsasuryy2.pdf?sequence=3&isAllowed=y

Sánchez Lanau, J. A. (2018). *Aplicaciones de las TIC en la ganadería bovina en el.* Barcelona: Universitat Politècnica de Catalunya.

Solidbi. (21 de 10 de 2021). *Solidbi. Inspira tu Innovación*. Obtenido de https://solid-bi.es/solidworks/

Sparkun. (s.f.). *SparkFun Pulse Oximeter and Heart Rate Sensor - MAX30101 & MAX32664 (Qwiic)*. Recuperado el octubre de 2021, de Description - Sparkun start something: https://www.sparkfun.com/products/15219

Vásconez, I. (2018). *Construcción e implementación de un robot móvil usando un SBC (Single-Board Computer) para aplicaciones de transporte de insumos médicos en el centro de salud tipo C Lizarzaburu en la ciudad de Riobamba.* Obtenido de Escuela Superior Politécnica de Chimborazo, Ecuador: http://dspace.espoch.edu.ec/bitstream/123456789/9240/1/108T0264.pdf

VERICAL. (2020). *TCRT5000.* VERICAL an Arrow Company.

Virgúez Javier. (04 de diciembre de 2016). *Microcontroladores*. Recuperado el octubre de 2021, de Universidad Fermín Toro. Facultad de Ingeniería. Escuela de Telecomunicaciones.: https://es.slideshare.net/javiervirguez/microcontroladores-69811192

ANEXOS

Anexo 1 - Programa principal

```
"""
Nombre: Programa principal, Robot Móvil para asistencia
Autor: Luis Manuel Benavente Salas y Frank Edward Chávez Aragón
luis.benanvente@ucsm.edu.pe        frank.chavez@ucsm.edu.pe
"""
from tkinter import *
from PIL import Image, ImageTk
import sensorMeasure as sm
import RPi.GPIO as GPIO
import numpy as np
import pandas as pd
import pygame, time, cv2, imutils
import seccionPreguntas
import arduinoComunicacion

def iniciar():
    """ Función para iniciar captura de vídeo"""
    global cap
    cap = cv2.VideoCapture(0)
    visualizar()

def visualizar():
    """ Función para mostrar vídeo en la interfaz de usuario """
    global cap
    # revisar si la variable cap contiene información
    if cap is not None:
        # obtener la imagen de la cámara
        ret, frame = cap.read()
        if ret == True:
            # redimensionar imagen a tamaño VGA
            frame = imutils.resize(frame, width=640)
            # convertir a espacio de color RGB
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            # convertir imagen tipo Array a tipo Image
            im = Image.fromarray(frame)
            img = ImageTk.PhotoImage(image=im)
            # mostrar vídeo
            lblVideo.configure(image=img)
            lblVideo.image = img
            lblVideo.after(10, visualizar)
        else:
            lblVideo.image = ""
            cap.release()

def finalizar():
    """ función para terminar captura de vídeo """
    global cap
    cap.release()

def bpmMeasure():
```

```
    """ funciones para obtener datos de sensores """
    # obtener valor de
    data = sm.getBpm()
    while data == None:
        print("repetir nuevamente")
        data = sm.getBpm()
    dataBpm = data[0]
    dataSpo2 = data[1]
    return dataBpm, dataSpo2

def tempMeasure():
    dataTemp = sm.getTemp()
    return dataTemp

def arteryPressMeasure():
    dataPress = sm.getArteryPressure()
    return dataPress

# función para cerrar ventana
def cerrar():
    root.destroy()

class contador():
    # constructor
    def __init__(self):
        self.cuenta = 1

    def incrementar(self):
        # incrementa el contador
        self.cuenta += 1

    def getConta(self):
        # regresar valor actual del contador
        return self.cuenta

conta = contador() # inicializar contador

cap = None
root = Tk()
root.geometry("1000x720") # tamaño de ventana
root.title('Robot Móvil') # título de ventana

# variables para la interfaz
temperature = StringVar()
bpm = StringVar()
spo2 = StringVar()
arteryPress = StringVar()
respTos = StringVar()
respRespiracion = StringVar()
respOlfato = StringVar()
respMalestar = StringVar()
respCefalea = StringVar()
respCongestión = StringVar()
respDiarrea = StringVar()
respGarganta = StringVar()
contadorPacientes = StringVar()
```

```
def salvarInfo(*args):
    preguntas = ["Tos", "Dificultad para respirar", "Dolor de
garganta", "Perdida de olfato",
                 "Malestar general", "Cefalea", "Congestión nasal",
"Diarrea", "Temperatura (°C)",
                 "Ritmo cardíaco (Bpm)", "Oxigenación (Spo2)",
"Presión arterial (Kpa)", "Fecha y hora"]
    fechaHora = time.strftime("%H:%M:%S") + ", " +
time.strftime("%d/%m/%y")
    respuestas = [respTos, respRespiracion, respGarganta,
respOlfato, respMalestar, respCefalea,
                  respCongestión, respDiarrea, temperature, bpm,
spo2, arteryPress, fechaHora]

    tabla = []

    for i in range(len(respuestas)):
        data = [preguntas[i], respuestas[i]]
        tabla.append(data)

    # salvar datos del paciente
    hora = time.strftime("%H:%M:%S")
    fecha = time.strftime("%d-%m-%y")
    numeroPaciente = str(conta.getConta())
    nombreArchivo = "paciente: " + numeroPaciente +" " + hora + "_"
+ fecha
    df = pd.DataFrame(tabla, columns = ['Preguntas', 'Respuestas'])
    df.to_csv("/home/pi/Desktop/Archivo                    de
pacientes/"+nombreArchivo+".csv")

def continuar(*args):
    # limpiar valor de variables
    time.sleep(1)
    moverContinuar() # poner en posición
    conta.incrementar() # aumentar el valor del contador en 1
    time.sleep(0.8)
    parar()

def estable(*args):
    # obtener temperatura
    pygame.mixer.init()
    pygame.mixer.music.load("/home/pi/Documents/Test
Sensors/audio/estable.mp3")
    pygame.mixer.music.set_volume(0.25)
    pygame.mixer.music.play()
    while pygame.mixer.music.get_busy() == True:
        continue

def noEstable(*args):
    # obtener temperatura
    pygame.mixer.init()

pygame.mixer.music.load("/home/pi/Documents/main/audio/noEstable.m
p3")
    pygame.mixer.music.set_volume(0.25)
```

```
    pygame.mixer.music.play()
    while pygame.mixer.music.get_busy() == True:
        continue

# botones principales
btnIniciar = Button(root, text= "Iniciar Vídeo", activeforeground =
'blue', width=10, command=iniciar)
btnIniciar.place(x=5, y=20)

btnFinalizar = Button(root, text= "Finalizar Vídeo",
activeforeground = 'red', width=10, command=finalizar)
btnFinalizar.place(x=150,y=20)

btnCerrar = Button(root, text = "Cerrar", activeforeground = 'red',
width=10, command=cerrar)
btnCerrar.place(x=300,y=20)

btnEstable = Button(root, text = "Estable", fg = 'black',
activeforeground = 'blue', width = 8, command = estable)
btnEstable.place(x = 670, y =590)

btnNoEstable = Button(root, text = "No Estable", activeforeground =
'red', width = 8, command = noEstable)
btnNoEstable.place(x = 770, y = 590)

btnSalvarInfo = Button(root, text = "Salvar Info", activeforeground
= 'blue', width = 8, command = salvarInfo)
btnSalvarInfo.place(x = 870, y = 590)

btnContinuar = Button(root, text = "Continuar", activeforeground =
'blue', width = 20, command = continuar)
btnContinuar.place(x=730, y = 640)

# Ventana de vídeo
lblVideo = Label(root)
lblVideo.place(x=5,y=60)

""" Función para realizar preguntas """
def quest(*args):
    pygame.mixer.init()

pygame.mixer.music.load("/home/pi/Documents/main/audio/intro.mp3")
    pygame.mixer.music.set_volume(0.25)
    pygame.mixer.music.play()
    while pygame.mixer.music.get_busy() == True:
        continue
    time.sleep(2)
    respuestas = seccionPreguntas.makeQuest()
    print('resp', respuestas)
    respTos.set(respuestas[0])
    respRespiracion.set(respuestas[1])
    respGarganta.set(respuestas[2])
    respOlfato.set(respuestas[3])
    respMalestar.set(respuestas[4])
    respCefalea.set(respuestas[5])
    respCongestión.set(respuestas[6])
```

```
    respDiarrea.set(respuestas[7])
    root.update()

""" Función para tomar signos vitales """
def task(*args):
    # obtener temperatura
import time,sys
sys.path.append('../')
import amg8833_i2c
import numpy as np


def getTemp():
    t0 = time.time()
    sensor = []
    while (time.time()-t0)<1: # esperar 1 segundo
        try:
            # AD0 = GND, addr = 0x68 | AD0 = 5V, addr = 0x69
            sensor = amg8833_i2c.AMG8833(addr=0x69) # inicilizar
AMG8833
        except:
            sensor = amg8833_i2c.AMG8833(addr=0x68)
        finally:
            pass
    time.sleep(0.2) # esperar establecer sensor
    temp = 0 # variable para almacenar valor de temperatura
    # Si no se encuentra sensor, regresar None
    if sensor==[]:
        print("No AMG8833 Found - Check Your Wiring")
        temp = None

    # Si se encuentra sensor, hacer toma de temperatura
    else:
        pix_to_read = 64 # read all 64 pixels
        temperatura = []
        for i in range(15):
            status,pixels = sensor.read_temp(pix_to_read) # read
pixels with status
            if status: # if error in pixel, re-enter loop and try
again
                continue

            T_thermistor = sensor.read_thermistor() # read
thermistor temp
            temperatura.append(max(pixels)) #print(max(pixels))
            time.sleep(1)
            print(temperatura)
    temp = np.mean(temperatura)
    return temp


#if __name__ == "__main__":
 #
  #  tempera = getTemp()
""" sección de cuestionario y respuestas """
```

```
btnQuiz = Button(root, text = "Iniciar Cuestionario",
activeforeground = 'blue', width = 15, command = quest)
btnQuiz.place(x = 660, y = 60)

lblQuiz1 = Label(root, text = "Tos: ", fg = 'black', font =
('Arial', 15))
lblQuiz1.place(x = 660, y = 130)
lblRespQuiz1 = Label(root, textvariable = respTos, fg = 'blue', font
= ('Arial', 15))
lblRespQuiz1.place(x = 780, y = 130)

lblQuiz2 = Label(root, text = "Dif. Resp.: ", fg = 'black', font =
('Arial', 15))
lblQuiz2.place(x = 660, y = 190)
lblRespQuiz2 = Label(root, textvariable = respRespiracion, fg =
'blue', font = ('Arial', 15))
lblRespQuiz2.place(x = 780, y = 190)

lblQuiz3 = Label(root, text = "Dolor Gar.: ", fg = 'black', font =
('Arial', 15))
lblQuiz3.place(x = 660, y = 240)
lblRespQuiz3 = Label(root, textvariable = respGarganta, fg = 'blue',
font = ('Arial', 15))
lblRespQuiz3.place(x = 780, y = 240)

lblQuiz4 = Label(root, text = "Perd. Olf.: ", fg = 'black', font =
('Arial', 15))
lblQuiz4.place(x = 660, y = 290)
lblRespQuiz4 = Label(root, textvariable = respOlfato, fg = 'blue',
font = ('Arial', 15))
lblRespQuiz4.place(x = 780, y = 290)

lblQuiz5 = Label(root, text = "Mal. Gen.: ", fg = 'black', font =
('Arial', 15))
lblQuiz5.place(x = 660, y = 340)
lblRespQuiz5 = Label(root, textvariable = respMalestar, fg = 'blue',
font = ('Arial', 15))
lblRespQuiz5.place(x = 780, y = 340)

lblQuiz6 = Label(root, text = "Cefalea: ", fg = 'black', font =
('Arial', 15))
lblQuiz6.place(x = 660, y = 390)
lblRespQuiz6 = Label(root, textvariable = respCefalea, fg = 'blue',
font = ('Arial', 15))
lblRespQuiz6.place(x = 780, y = 390)

lblQuiz7 = Label(root, text = "Cong. Nas.: ", fg = 'black', font =
('Arial', 15))
lblQuiz7.place(x = 660, y = 440)
lblRespQuiz7 = Label(root, textvariable = respCongestión, fg =
'blue', font = ('Arial', 15))
lblRespQuiz7.place(x = 780, y = 440)

lblQuiz8 = Label(root, text = "Diarrea: ", fg = 'black', font =
('Arial', 15))
lblQuiz8.place(x = 660, y = 490) # 820
```

```
lblRespQuiz8 = Label(root, textvariable = respDiarrea, fg = 'blue',
font = ('Arial', 15))
lblRespQuiz8.place(x = 780, y = 490)

""" botón para obtener sensado e indicadores """
# botón iniciar sensado
btnTask = Button(root, text="Iniciar sensado", activeforeground =
'blue', width = 10, command = task)
btnTask.place(x=855, y = 60)
# label de temperatura
lblTitleTemp = Label(root, text = 'Temperatura', fg = 'red', font =
('Arial', 15))
lblTitleTemp.place(x=855, y = 130) #150
lblTemp = Label(root, textvariable = temperature, fg = 'red', font
= ('Arial', 25))
lblTemp.place(x=855,y = 170)
lblTextTemp = Label(root, text = '°C', fg = 'red', font = ('Arial',
15))
lblTextTemp.place(x=945, y =180)
# label BPM
lblTitleBpm = Label(root, text = 'Ritmo Cardiaco', fg = 'dark
orange', font = ('Arial',15))
lblTitleBpm.place(x=855, y = 240)
lblBpm = Label(root, textvariable = bpm, fg = 'dark orange', font =
('Arial', 25))
lblBpm.place(x=855, y = 280)
lblTextBpm = Label(root, text = 'BPM', fg = 'dark orange', font =
('Arial', 15))
lblTextBpm.place(x = 940, y =290)
# label SPO2
lblTitleSpo2 = Label(root, text = 'Oxigenación', fg = 'dodger blue',
font = ('Arial', 15))
lblTitleSpo2.place(x = 885, y = 350)
lblSpo2 = Label(root, textvariable = spo2, fg = 'dodger blue', font
= ('Arial', 25))
lblSpo2.place(x = 855, y = 380)
lblTextSpo2 = Label(root, text = 'SPO2', fg = 'dodger blue', font =
('Arial', 15))
lblTextSpo2.place(x = 935, y = 400)
# label Presión arterial
lblTitlePress = Label(root, text = 'Presión Arterial', fg = 'green',
font = ('Arial', 15))
lblTitlePress.place(x = 855, y = 460)
lblPress = Label(root, textvariable = arteryPress, fg = 'green',
font = ('Arial', 25))
lblPress.place(x = 855, y = 490)
lblTextPress = Label(root, text = 'KPa', fg = 'green', font =
('Arial', 15))
lblTextPress.place(x = 940, y = 510)

""" Sección para control manual """
def moverRobot(giro):
    arduinoComunicación.moveRobot(giro)

def moverDerecha(*args):
    giro = "3"
```

```
    moverRobot(giro)

def moverIzquierda(*args):
    giro = "4"
    moverRobot(giro)

def moverAdelante(*args):
    giro = "2"
    moverRobot(giro)

def moverAtrás(*args):
    giro = "5"
    moverRobot(giro)

def parar(*args):
    giro = "6"
    moverRobot(giro)

def moverAutomático(*args):
    giro = "1"
    moverRobot(giro)

def moverContinuar(*args):
    giro = "7"
    moverRobot(giro)

# indicador - label
lblManual = Label(root, text = "Control Manual", fg = 'Black', font
= ('Arial', 16))
lblManual.place(x = 130, y =550)

btnGiroIzquierda = Button(root, text = 'Izquierda', activebackground
= 'steelBlue4', activeforeground = 'white', width = 10, command =
moverIzquierda)
btnGiroIzquierda.place(x = 5, y = 640)
# botón adelante
btnGiroAdelante = Button(root, text = 'Adelante', activebackground
= 'steelBlue4', activeforeground = 'white', width = 10, command =
moverAdelante)
btnGiroAdelante.place(x = 140, y = 600)
# botón derecha
btnGiroDerecha = Button(root, text = 'Derecha', activebackground =
'steelBlue4', activeforeground = 'white', width = 10, command =
moverDerecha)
btnGiroDerecha.place(x = 275, y = 640)
# botón atrás
btnGiroAtrás = Button(root, text = 'Atrás', activebackground =
'steelBlue4', activeforeground = 'white', width = 10, command =
moverAtrás)
btnGiroAtrás.place(x = 140, y = 680)
# botón parar
btnParar = Button(root, text = 'Parar', activebackground = 'red2',
activeforeground = 'white', width = 10, command = parar)
btnParar.place(x = 140, y = 640)

""" Sección para el control automático """
```

```
lblAutomático = Label(root, text = "Control Automático", fg =
"Black", font = ('Arial', 16))
lblAutomático.place(x = 440, y = 550)
# botón de activación de modo automático
btnAutomático = Button(root, text = "Automático", activebackground
= 'steelBlue4', activeforeground = 'white', width = 10, command =
moverAutomático)
btnAutomático.place(x = 480, y = 640)

root.mainloop()
```

Anexo 2 – Codificación para la detección de guía azul y reconocimiento facial

```
      #  detectar conos azules
                        _, direccion = pdi.track(frameTrack)
                        serialData = 'No'
                        if direccion == 'C':
                            # mover al centro
                            arduino.flushInput()
                            arduino.write(b'2')
                            serialData = arduino.readline().decode('utf-
8').rstrip()
                            print('serialData: ', serialData)
                        elif direccion == 'R':
                            # mover a la derecha
                            arduino.flushInput()
                            arduino.write(b'3')
                            serialData = arduino.readline().decode('utf-
8').rstrip()
                            print('serialData: ', serialData)
                        elif direccion == 'L':
                            # mover a la izquieda
                            arduino.flushInput()
                            arduino.write(b'4')
                            serialData = arduino.readline().decode('utf-
8').rstrip()
                            print('serialData: ', serialData)

                        # algoritmo evasor de obstáculos
                        if serialData == "Obstaculo":
      # si detectó obstaculo, identificar si es una persona por
detección de rostro
                            if posx == 0:
                                # no se detectó rostro, evadir obstáculo
                                arduino.write(b'9') # mover atras
                                time.sleep(1)
                                arduino.write("A".encode())    #    mover
izquierda 90°
                                time.sleep(1)
                            elif posx > 0:
                                # actualizar interfaz
                                # borrarDatos()
                                #  si  se  detectó  rostro,  atender  al
paciente
                                routinePatient()  #  aplicar  rutina  de
atención al paciente
                                # una vez terminado la rutina, evadir
persona
                                arduino.write(b'9') # mover atras
                                time.sleep(1)
                                arduino.write("A".encode())    #    mover
izquierda 45°
                                time.sleep(20) # 20 segundos de esperas
                                continuar()  #  continuar  busqueda  de
pacientes
                        # si no se detectó un rosto, ni objeto de color
de interés,
                        # girar a la izquierda para buscar un rostro o un
objeto de interés
```

```
                    if serialData == 'No':
                        arduino.flushInput()
                        arduino.write(b'4') # giro a la izquierda
                    lblVideo.after(10, visualizar) # regresar a la
función para refrescar video
                else:
                    # si no se activó el modo autónomo, regresar a la
función para refrescar video
                    lblVideo.after(10, visualizar)
            # si se desactiva la cámara, no mostrar video y terminar
conexión con cámaras
            else:

                lblVideo.image = ""
                cameraFace.release()
                cameraTrack.release()


    def routinePatient():
        global buscar
        global inconciente
        inconciente = False
        state = 1
        while state < 2:
            # generar cuestionario
            quest()
            time.sleep(2)
            if inconciente == True:
                # toma de temperatura
                temp()
            elif inconciente == False:
                # toma de signos vitales
                task()
                time.sleep(1)
            state +=1
        parar()
        buscar = False
```

Anexo 3- Codificación para obtener saturación de oxígeno y ritmo cardíaco

```
"""
Nombre: biblioteca para establecer comunicación y obtener datos del
sensor MAX30100
Autor: Luis Manuel Benavente Salas y Frank Edward Chávez Aragón
luis.benanvente@ucsm.edu.pe          frank.chavez@ucsm.edu.pe
"""
# biblioteca para comunicación i2c
import smbus

INT_STATUS    = 0x00
INT_ENABLE    = 0x01
FIFO_WR_PTR   = 0x02
OVRFLOW_CTR   = 0x03
FIFO_RD_PTR   = 0x04
FIFO_DATA     = 0x05
MODE_CONFIG   = 0x06
SPO2_CONFIG   = 0x07
LED_CONFIG    = 0x09
TEMP_INTG     = 0x16
TEMP_FRAC     = 0x17
REV_ID        = 0xFE
PART_ID       = 0xFF
I2C_ADDRESS   = 0x57

PULSE_WIDTH = {
    200: 0,
    400: 1,
    800: 2,
   1600: 3,
}

SAMPLE_RATE = {
    50: 0,
   100: 1,
   167: 2,
   200: 3,
   400: 4,
   600: 5,
   800: 6,
  1000: 7,
}

LED_CURRENT = {
       0: 0,
     4.4: 1,
     7.6: 2,
    11.0: 3,
    14.2: 4,
    17.4: 5,
    20.8: 6,
    24.0: 7,
    27.1: 8,
    30.6: 9,
    33.8: 10,
```

```
    37.0: 11,
    40.2: 12,
    43.6: 13,
    46.8: 14,
    50.0: 15
}

def _get_valid(d, value):
    try:
        return d[value]
    except KeyError:
        raise KeyError("Value %s not valid, use one of: %s" % (value,
', '.join([str(s) for s in d.keys()])))

def _twos_complement(val, bits):

    if (val & (1 << (bits - 1))) != 0: # if sign bit is set e.g.,
8bit: 128-255
        val = val - (1 << bits)
    return val

INTERRUPT_SPO2 = 0
INTERRUPT_HR = 1
INTERRUPT_TEMP = 2
INTERRUPT_FIFO = 3

MODE_HR = 0x02
MODE_SPO2 = 0x03

class MAX30100(object):

    def __init__(self,
                 i2c=None,
                 mode=MODE_HR,
                 sample_rate=100,
                 led_current_red=11.0,
                 led_current_ir=11.0,
                 pulse_width=1600,
                 max_buffer_len=10000
                 ):


        self.i2c = i2c if i2c else smbus.SMBus(1)

        self.set_mode(MODE_HR)   # Trigger an initial temperature
read.
        self.set_led_current(led_current_red, led_current_ir)
        self.set_spo_config(sample_rate, pulse_width)


        self.buffer_red = []
        self.buffer_ir = []

        self.max_buffer_len = max_buffer_len
        self._interrupt = None
```

```
    @property
    def red(self):
        return self.buffer_red[-1] if self.buffer_red else None

    @property
    def ir(self):
        return self.buffer_ir[-1] if self.buffer_ir else None

    def set_led_current(self, led_current_red=11.0,
led_current_ir=11.0):

        led_current_red = _get_valid(LED_CURRENT, led_current_red)
        led_current_ir = _get_valid(LED_CURRENT, led_current_ir)
        self.i2c.write_byte_data(I2C_ADDRESS, LED_CONFIG,
(led_current_red << 4) | led_current_ir)

    def set_mode(self, mode):
        reg = self.i2c.read_byte_data(I2C_ADDRESS, MODE_CONFIG)
        self.i2c.write_byte_data(I2C_ADDRESS, MODE_CONFIG, reg &
0x74) # mask the SHDN bit
        self.i2c.write_byte_data(I2C_ADDRESS, MODE_CONFIG, reg |
mode)

    def set_spo_config(self, sample_rate=100, pulse_width=1600):
        reg = self.i2c.read_byte_data(I2C_ADDRESS, SPO2_CONFIG)
        reg = reg & 0xFC  # Set LED pulsewidth to 00
        self.i2c.write_byte_data(I2C_ADDRESS, SPO2_CONFIG, reg |
pulse_width)

    def enable_spo2(self):
        self.set_mode(MODE_SPO2)

    def disable_spo2(self):
        self.set_mode(MODE_HR)

    def enable_interrupt(self, interrupt_type):
        self.i2c.write_byte_data(I2C_ADDRESS, INT_ENABLE,
(interrupt_type + 1)<<4)
        self.i2c.read_byte_data(I2C_ADDRESS, INT_STATUS)

    def get_number_of_samples(self):
        write_ptr = self.i2c.read_byte_data(I2C_ADDRESS,
FIFO_WR_PTR)
        read_ptr = self.i2c.read_byte_data(I2C_ADDRESS,
FIFO_RD_PTR)
        return abs(16+write_ptr - read_ptr) % 16

    def read_sensor(self):
        bytes = self.i2c.read_i2c_block_data(I2C_ADDRESS,
FIFO_DATA, 4)

        self.buffer_ir.append(bytes[0]<<8 | bytes[1])
        self.buffer_red.append(bytes[2]<<8 | bytes[3])

        self.buffer_red = self.buffer_red[-self.max_buffer_len:]
        self.buffer_ir = self.buffer_ir[-self.max_buffer_len:]
```

```
    def shutdown(self):
        reg = self.i2c.read_byte_data(I2C_ADDRESS, MODE_CONFIG)
        self.i2c.write_byte_data(I2C_ADDRESS,  MODE_CONFIG,  reg  |
0x80)

    def reset(self):
        reg = self.i2c.read_byte_data(I2C_ADDRESS, MODE_CONFIG)
        self.i2c.write_byte_data(I2C_ADDRESS,  MODE_CONFIG,  reg  |
0x40)

    def refresh_temperature(self):
        reg = self.i2c.read_byte_data(I2C_ADDRESS, MODE_CONFIG)
        self.i2c.write_byte_data(I2C_ADDRESS, MODE_CONFIG, reg | (1
<< 3))

    def get_temperature(self):
        intg                                                    =
_twos_complement(self.i2c.read_byte_data(I2C_ADDRESS, TEMP_INTG))
        frac = self.i2c.read_byte_data(I2C_ADDRESS, TEMP_FRAC)
        return intg + (frac * 0.0625)

    def get_rev_id(self):
        return self.i2c.read_byte_data(I2C_ADDRESS, REV_ID)

    def get_part_id(self):
        return self.i2c.read_byte_data(I2C_ADDRESS, PART_ID)

    def get_registers(self):
        return {
            "INT_STATUS":     self.i2c.read_byte_data(I2C_ADDRESS,
INT_STATUS),
            "INT_ENABLE":     self.i2c.read_byte_data(I2C_ADDRESS,
INT_ENABLE),
            "FIFO_WR_PTR":    self.i2c.read_byte_data(I2C_ADDRESS,
FIFO_WR_PTR),
            "OVRFLOW_CTR":    self.i2c.read_byte_data(I2C_ADDRESS,
OVRFLOW_CTR),
            "FIFO_RD_PTR":    self.i2c.read_byte_data(I2C_ADDRESS,
FIFO_RD_PTR),
            "FIFO_DATA":      self.i2c.read_byte_data(I2C_ADDRESS,
FIFO_DATA),
            "MODE_CONFIG":    self.i2c.read_byte_data(I2C_ADDRESS,
MODE_CONFIG),
            "SPO2_CONFIG":    self.i2c.read_byte_data(I2C_ADDRESS,
SPO2_CONFIG),
            "LED_CONFIG":     self.i2c.read_byte_data(I2C_ADDRESS,
LED_CONFIG),
            "TEMP_INTG":      self.i2c.read_byte_data(I2C_ADDRESS,
TEMP_INTG),
            "TEMP_FRAC":      self.i2c.read_byte_data(I2C_ADDRESS,
TEMP_FRAC),
            "REV_ID": self.i2c.read_byte_data(I2C_ADDRESS, REV_ID),
            "PART_ID":        self.i2c.read_byte_data(I2C_ADDRESS,
PART_ID),
        }
```

Anexo 4 - Codificación para obtener la presión arterial

```
"""
Nombre: Lectura de sensor MPX10DP por MCP3008
Autor: Luis Manuel Benavente Salas y Frank Edward Chávez Aragón
luis.benanvente@ucsm.edu.pe        frank.chavez@ucsm.edu.pe
"""
import spidev
import time
import os

# Inicializar comunicación SPI
spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz = 1000000

def ReadChannel(channel):
    """ Función para leer datos SPI desde el integrado MCP3008 """
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data = ((adc[1]&3) << 8) + adc[2]
  return data

def ConvertVolts(data,places):
  """ Función para convertir datos a nivel de voltaje """
  volts = (data * 3.3) / float(1023)
  volts = round(volts,places)
  return volts

# canal donde se ha conectado el sensor MPX10DP
mpxChannel = 0
sensorOffset = 102.0

# Retardo entre lecturas
delay = 0.5

while True:

  # Leer datos del sensor
  sensorValue = ReadChannel(mpxChannel)
  # convertir señal digital a señal de presión
  pressure = (sensorValue - sensorOffset) / 100.0
  # mostrar datos en el terminal
  print("Value: {} ({} kPa)".format(sensorValue,pressure))

  # esperar
  time.sleep(delay)
```

Anexo 5 - Codificación para la toma de datos vitales del paciente (incluyendo temperatura)

```
"""
Nombre: Biblioteca de funciones para lectura de sensores
Autor: Luis Manuel Benavente Salas y Frank Edward Chávez Aragón
luis.benanvente@ucsm.edu.pe        frank.chavez@ucsm.edu.pe
"""
import RPi.GPIO as GPIO
import time
import max30100
import numpy as np
import board
import busio as io
import adafruit_mlx90614
from time import sleep
import spidev
import os

""" Funciones para lectura de oxímetro y ritmo cardíaco """
def mover_promedio(numbers):
    """ función para obtener el promedio de lecturas del sensor de
ritmo cardíaco """
    window_size = 4
    i = 0
    while i < len(numbers) - window_size + 1:
        this_window = numbers[i : i + window_size]
        window_average = sum(this_window) / window_size
        i += 1
    try:
        return int((window_average/100))
    except:
        pass

def display_filter(moving_average_bpm,moving_average_sp02):
    """ Función para aplicar un filtro a los datos del sensor """
    # Si el bpm es <10 la función asume que no ha encontrado datos
presentes y no mostrara datos incorrectos
    # también si las lecturas de SpO2  son superiores a 100. Las
Mostrará Como 100
    try:
        if(moving_average_bpm<10):
            moving_average_bpm ='NA'
            moving_average_sp02 = 'NA'
        else:
            if(moving_average_sp02>100):
                moving_average_sp02 = 100
        return moving_average_bpm, moving_average_sp02
    except:
        return moving_average_bpm, moving_average_sp02

# get measure
def getBpm():
    """  Función  para  obtener  lecturas  de  ritmo  cardíaco  y
oxigenación en la sangre vía sensor MAX 30100
```

```
        Regresa: bpm, spo2: lista de dos elementos escalares tipo
float
    """
    mx30 = max30100.MAX30100()
    mx30.enable_spo2()# Generates Moving Average - This will filter
data and improve stability of readings
    print('colocar dedo sobre sensor...')
    time.sleep(3)
    listBpm = [] # lista para almacenar datos de ritmo cardíaco
    listSpo2 = [] # lista para almacenar datos de oxigenación
    numSample = 10
    for i in range(numSample):
        # tomar lecturas del sensor
        mx30.read_sensor()
        hb = int(mx30.ir / 100) # lectura de ritmo cardíaco
        spo2 = int(mx30.red / 100) # lectura de oxigenación en la
sangre
        if mx30.ir != mx30.buffer_ir:
            mover_promedio_bpm = (mover_promedio(mx30.buffer_ir))
        if mx30.red != mx30.buffer_red:
            mover_promedio_sp02 = (mover_promedio(mx30.buffer_red))

        # aplicar un filtro a los datos obtenidos
        bpm,spo2                                                =
display_filter(mover_promedio_bpm,mover_promedio_sp02)
        listBpm.append(bpm) # agregar el dato de bpm a la lista
listBpm
        listSpo2.append(spo2) # agregar el dato de spo2 a la lista
listSpo2
        # mostrar los datos en el terminal
        print(" ********************")
        print(" bpm : "+ str(bpm))
        print(" SpO2: "+ str(spo2))
        print(" --------------------")
        time.sleep(1)
    # acondicionar datos
    dataBpm = np.array(listBpm) # convertir listas a arreglos
    dataSpo2 = np.array(listSpo2)
    noData = np.where(dataBpm == "NA") # identificar datos sin
información (NA)
    numSample = numSample - noData[0].shape[0] # restar a 10 el
valor de datos sin información
    dataBpm = np.delete(dataBpm, noData) # eliminar datos sin
información
    dataSpo2 = np.delete(dataSpo2, noData) # eliminar datos sin
información
    f = np.where(dataBpm == None) # identificar datos sin algún
valor (None)
    if f[0].shape[0] < numSample:
        dataBpm = np.delete(dataBpm, f) # eliminar datos sin
información
        dataSpo2 = np.delete(dataSpo2, f) # eliminar datos sin
información
        outBpm = np.median(dataBpm) # aplicar el valor de la mediana
a la lista de datos bpm
```

```
        outSpo2 = np.median(dataSpo2) # aplicar el valor de la
mediana a la lista de datos spo2
        out = [outBpm, outSpo2] # salida de la función
    elif f[0].shape[0] == numSample:
        # en caso contrario la función tendrá el valor None
        out = None

    return out

def getTemp():
    """ Función para obtener temperatura del módulo AMG-8833 """
    #obtener datos de temperatura del módulo AMG-8833
import time,sys
sys.path.append('../')
import amg8833_i2c
import numpy as np


def getTemp():
    t0 = time.time()
    sensor = []
    while (time.time()-t0)<1: # esperar 1 segundo
        try:
            # AD0 = GND, addr = 0x68 | AD0 = 5V, addr = 0x69
            sensor = amg8833_i2c.AMG8833(addr=0x69) # inicializar
AMG8833
        except:
            sensor = amg8833_i2c.AMG8833(addr=0x68)
        finally:
            pass
    time.sleep(0.2) # esperar establecer sensor
    temp = 0 # variable para almacenar valor de temperatura
    # Si no se encuentra sensor, regresar None
    if sensor==[]:
        print("No AMG8833 Found - Check Your Wiring")
        temp = None

    # Si se encuentra sensor, hacer toma de temperatura
    else:
        pix_to_read = 64 # read all 64 pixels
        temperatura = []
        for i in range(15):
            status,pixels = sensor.read_temp(pix_to_read) # read
pixels with status
            if status: # if error in pixel, re-enter loop and try
again
                continue

            T_thermistor = sensor.read_thermistor() # read
thermistor temp
            temperatura.append(max(pixels)) #print(max(pixels))
            time.sleep(1)
            print(temperatura)
    temp = np.mean(temperatura)
    return temp
```

```
#if __name__ == "__main__":
 #
  #  tempera = getTemp()
""" Funciones para obtener presión arterial por MPX10DP, por medio
del integrado """
# Abrir bus SPI
spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz=1000000

def ReadChannel(channel):
    """ Función para leer datos por SPI, desde el integrado MCP3008
        Entrada: Channel -> deber ser un entero, entre 0 y 7
        Salida: data -> valor escalar, float
    """
    adc  =  spi.xfer2([1,(8+channel)<<4,0])  #  seleccionar  canal
analógico
    data = ((adc[1]&3) << 8) + adc[2] # leer canal analógico
    return data

def getArteryPressure():
    """ Función para medir presión arterial
        Regresa: Valor medio (escalar, tipo float), obtenido de las
lecturas de presión
    """
    pinMotor = 12 #32 # puerto para conectar señal de activación
del motor
    pinValve = 13 #33 # puerto para conectar señal de activación de
la electroválvula
    # configuración de puertos
    GPIO.setmode(GPIO.BCM)  # leer  puertos,  en  base  a  como  el
microprocesador los define
    GPIO.setup(pinMotor, GPIO.OUT) # establecer puerto como salida
    GPIO.setup(pinValve, GPIO.OUT) # establecer puerto como salida
    # canal de conexión del sensor MPX10DP
    mpxChannel = 0
    # offset
    sensorOffset = 102.0
    # Retardo entre lecturas
    delay = 0.5
    # Comenzar con la lectura
    print("Colocar tensiómetro")
    dataPress = []
    # inicia rutina para medir presión arterial
    for i in range(5):
        # activar bomba
        time.sleep(2)
        GPIO.output(pinMotor, True) # activa motor de bomba de aire
        GPIO.output(pinValve, False) # desactiva electroválvula
        time.sleep(7) # activar bomba de aire por 7 segundos
        # activar electroválvula
        GPIO.output(pinMotor, False) # desactivar bomba de aire
        GPIO.output(pinValve, True) # activa electroválvula
        time.sleep(0.1) # esperar 10 ms
        # Leer datos del sensor
```

```
        sensorValue = ReadChannel(mpxChannel) # leer canal 0 del
módulo MCP, para obtener lecturas del sensor de presión de aire
        pressure = (sensorValue - sensorOffset) / 100.0 # convertir
valor analógico obtenido del sensor, a dato de presión en Kpa
        dataPress.append(pressure) # almacenar
        # Print out results
        print("Value: {} ({} kPa)".format(sensorValue,pressure))
        # Wait before repeating loop
        time.sleep(delay)
    # apagar/desactivar motor y electroválvula
    GPIO.output(pinMotor, False)
    GPIO.output(pinValve, False)
    time.sleep(2)
    # limpiar puertos de la Rpi
    GPIO.cleanup()
    return np.mean(dataPress) # regresar valor medio obtenido de las
lecturas de presión
```

Anexo 6 – Codificación para la programación del cuestionario

```
"""
Nombre: Función para hacer una pregunta
Autor: Luis Manuel Benavente Salas y Frank Edward Chávez Aragón
luis.benanvente@ucsm.edu.pe        frank.chavez@ucsm.edu.pe
"""
import RPi.GPIO as GPIO
import time
import numpy as np
import pandas as pd
import csv
import pygame

def preguntar (pregunta, respuestas):
    """ Función para hacer una pegunta y recibir la respuesta
respectiva
        Recibe: Pregunta -> String con el nombre de la pregunta
        Regresa: Lista  String -> Lista de respuestas en String
    """
    greenButton = 17
    redButton = 27
    yellowButton = 22
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(greenButton, GPIO.IN, pull_up_down = GPIO.PUD_UP)
    GPIO.setup(redButton, GPIO.IN, pull_up_down = GPIO.PUD_UP)
    GPIO.setup(yellowButton, GPIO.IN, pull_up_down = GPIO.PUD_UP)
    # generar pregunta
    pygame.mixer.init()

pygame.mixer.music.load("/home/pi/Documents/main/audio/"+pregunta)
    pygame.mixer.music.set_volume(0.25)
    pygame.mixer.music.play()

    while True:
        respSi = GPIO.input(greenButton)
        respNo = GPIO.input(redButton)
        respNa = GPIO.input(yellowButton)
        if respSi == 0:
            print('Si: ', respSi)
            respuesta = 'Si'
            respuestas.append(respuesta)
            time.sleep(0.3)
            break
        if respNo == 0:
            print('No: ', respNo)
            respuesta = 'No'
            respuestas.append(respuesta)
            time.sleep(0.3)
            break
        if respNa == 0:
            print('Na: ', respNa)
            respuesta = 'Na'
            respuestas.append(respuesta)
            time.sleep(0.3)
            break
```

```
            time.sleep(0.2)
        time.sleep(1)
        GPIO.cleanup()
        return respuestas

def makeQuest():
    """ Función para hacer preguntas del cuestionario """
    # lista de preguntas
    preguntas                                                    =
['tos.mp3','respiracion.mp3','garganta.mp3','olfato.mp3',
                'malestar.mp3','cefalea.mp3','congestion.mp3',
'diarrea.mp3']
    respuestas = []
    for i in range(len(preguntas)):
        respuestas = preguntar(preguntas[i], respuestas)
    return respuestas

if __name__ == '__main__':
    # generar intro
    pygame.mixer.init()

pygame.mixer.music.load("/home/pi/Documents/main/audio/intro.mp3")
    pygame.mixer.music.set_volume(0.25)
    pygame.mixer.music.play()
    while pygame.mixer.music.get_busy() == True:
        continue
    time.sleep(2)
    respuestas = makeQuest()

    df = pd.DataFrame(respuestas)
    df.to_csv('/home/pi/resp.csv')
```

Anexo 7 - Codificación para comunicación con ATMega328

```
"""
Nombre: Comunicación con ATMega328 vía serial
Autor: Luis Manuel Benavente Salas y Frank Edward Chávez Aragón
luis.benanvente@ucsm.edu.pe        frank.chavez@ucsm.edu.pe
"""
import serial
import time

def moveRobot(comando = "6"):
    """ Función para enviar un comando vía serial
    entrada:
        comando : tipo string
        descripción: comando que se desea enviar vía serial
    """
    with serial.Serial("/dev/ttyACM0", 9600, timeout=1) as arduino:
        time.sleep(0.1)   #   esperar   para   conectar   con   el
microcontrolador
        if arduino.isOpen():
            print("{} conectado!".format(arduino.port))
            try:
                time.sleep(2)
                arduino.write(comando.encode())
            except KeyboardInterrupt:
                print(".")
```

Anexo 8 - Codificación del desplazamiento del robot móvil (ATMega328)

```
/*
 * Nombre: Comunicación ATMega328 - Raspberry Pi 4, para control de movimiento de robot Móvil
 * Autor: Luis Manuel Benavente Salas y Frank Edward Chávez Aragón
luis.benanvente@ucsm.edu.pe        frank.chavez@ucsm.edu.pe
 */
// Incluir bibliotecas
#include "sonar.h"
#include "serialComunication.h"

// variable para almacenar comando serial
char data;

void setup() {
  pinMode(s1, INPUT);
  pinMode(s2, INPUT);
  pinMode(s3, INPUT);
  pinMode(enab1, OUTPUT);
  pinMode(enab2, OUTPUT);
  pinMode(motor1, OUTPUT);
  pinMode(motor2, OUTPUT);
  pinMode(motor3, OUTPUT);
  pinMode(motor4, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // leer serial
  readSerial();
  checkObstacle();
  // variables de distancia
  if (serialData == '1')
  {
    automático();
    data = serialData;
  }

  if (serialData == '2')
  {
    // Mover adelante
    if (obstaculoFrente == 0)
    {
      giroAdelante();
      data = serialData;
      delay(2);
    }
    else
    {
      parar();
      Serial.println("Obstáculo al Frente");
    }
  }
  if (serialData == '3')
```

```
{
  if (obstáculoFrente == 0)
  {
    // giro derecha
    giroDer();
    data = serialData;
    delay(2);
  }
  else
  {
    parar();
    Serial.println("Obstaculo al Frente");
  }
}

if (serialData == '4')
{
  if (obstaculoFrente == 0)
  {
    giroIzq();
    data = serialData;
    delay(2);
  }
  else
  {
    parar();
    Serial.println("Obstaculo al Frente");
  }
}

if (serialData == '5')
{
  if (obstáculoAtrás == 0)
  {
    giroAtrás();
    data = serialData;
    delay(2);
  }
  else
  {
    parar();
    Serial.println("Obstaculo atrás");
  }
}

if (serialData == '6')
{
  parar();
}

if (serialData == '7')
{
  continuar();
}
  // si hay obstáculo al frente, parar giro adelante,
izquierdo y derecho
```

```
        if ((obstáculoFrente == 1) & ((data == '2') || (data == '3')
|| (data == '4')))
        {
          parar();
        }
        // si hay obstáculo atrás, para giro atrás
        if ((obstáculoAtras == 1) & (data == '5'))
        {
          parar();
        }
      }
```

Anexo 9 - Codificación para la comunicación serial

```
/*
 * Nombre: Biblioteca de funciones para comunicación serial
 * Autor: Luis Manuel Benavente Salas y Frank Edward Chávez
Aragón
luis.benanvente@ucsm.edu.pe        frank.chavez@ucsm.edu.pe
 */
// variable para almacenar información serial
char serialData;

void readSerial()
{
  // función para leer comandos del puerto serial
  if (Serial.available())
  {
    serialData = Serial.read();
  }
}
void continuar()
{
  // Función para continuar el seguimiento de línea, después
de sensado
  giroDer();
  delay(100);
  parar();
}
```

Anexo 10 - Codificación del sensor de detección de obstáculos

```
/*
 * Autor: Luis Manuel Benavente Salas y Frank Edward Chávez Aragón
luis.benanvente@ucsm.edu.pe        frank.chavez@ucsm.edu.pe
 */

#include <NewPing.h>

#define SONAR_NUM 4      // Número de sensores
#define MAX_DISTANCE 300 // Distancia máxima en cm

NewPing sonar[SONAR_NUM] = {
  NewPing(A4, A5, MAX_DISTANCE), // trigger pin, echo pin
  NewPing(8, 9, MAX_DISTANCE),
  NewPing(11, 10, MAX_DISTANCE),
  NewPing(12, A3, MAX_DISTANCE)
};

// variables de distancia
int distPingFrontR;
int distPingFrontL;
int distPingBackR;
int distPingBackL;
int obstaculoFrente = 0;
int obstaculoAtras = 0;

// umbral para detectar objetos
int umbralFrente = 30;
int umbralAtras = 15;

void checkObstacle()
{
  // sonar
  distPingFrontR = sonar[0].ping_cm();
  delay(10);
  distPingFrontL = sonar[1].ping_cm();
  delay(10);
  distPingBackR = sonar[2].ping_cm();
  delay(10);
  distPingBackL = sonar[3].ping_cm();
  delay(10);
  if (((distPingFrontR != 0) & (distPingFrontR <= umbralFrente)) ||
((distPingFrontL != 0) & (distPingFrontL <= umbralFrente)))
  {
    obstáculoFrente = 1;
  }
  else
  {
    obstáculoFrente = 0;
  }

  if (((distPingBackR != 0) & (distPingBackR < umbralAtrás)) ||
((distPingBackL != 0) & (distPingBackL < umbralAtrás)))
  {
```

```
      obstaculoAtras = 1;
    }
    else
    {
      obstaculoAtras = 0;
    }
}
```

Anexo 11 - Codificación para las alarmas de fuera de rango

```
      /*
       * Autor: Luis Manuel Benavente Salas y Frank Edward Chávez
Aragón
      luis.benanvente@ucsm.edu.pe        frank.chavez@ucsm.edu.pe
       */

# analizar umbrales de signos vitales
    if valTempNum < 38.5 and valTempNum > 35:
    #   reproducir temperatura normal
        pygame.mixer.init()
        pygame.mixer.music.load("/home/pi/Documents/Codigo
Principal/Principal/audio/temperatura_normal.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    if valTempNum > 38.5:
    #   indicar temperatura alta
        pygame.mixer.init()
        pygame.mixer.music.load("/home/pi/Documents/Codigo
Principal/Principal/audio/temperatura_alta.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    if valTempNum < 35:
        pygame.mixer.init()

pygame.mixer.music.load("/home/pi/Documents/Principal_v2/audio/temper
atura_baja.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    if valBpm < 60:
    #   indicar ritmo cardiaco bajo
        pygame.mixer.init()
        pygame.mixer.music.load("/home/pi/Documents/Codigo
Principal/Principal/audio/bpm_bajo.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    if valBpm > 130:
    #   indicar ritmo cardiaco alto
        pygame.mixer.init()
        pygame.mixer.music.load("/home/pi/Documents/Codigo
Principal/Principal/audio/bpm_alto.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    if (valBpm > 60) and (valBpm < 130):
    #   indicar ritmo cardiaco estable
        pygame.mixer.init()
```

```
        pygame.mixer.music.load("/home/pi/Documents/Codigo
Principal/Principal/audio/bpm_estable.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    if valSpo2 < 94:
    #   indicar saturación de oxigeno baja
        pygame.mixer.init()
        pygame.mixer.music.load("/home/pi/Documents/Codigo
Principal/Principal/audio/oxigenacion_baja.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    if valSpo2 >=94:
    #   indicar oxigenación normal
        pygame.mixer.init()
        pygame.mixer.music.load("/home/pi/Documents/Codigo
Principal/Principal/audio/oxigenacion_normal.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    if valPress < 7:
    #   indicar presión arterial baja
        pygame.mixer.init()
        pygame.mixer.music.load("/home/pi/Documents/Codigo
Principal/Principal/audio/presion_baja.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    if valPress > 12:
    #   indicar presión artrial alta
        pygame.mixer.init()
        pygame.mixer.music.load("/home/pi/Documents/Codigo
Principal/Principal/audio/presion_alta.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    if (valPress > 7) and (valPress < 12):
    #   indicar presión arterial normal
        pygame.mixer.init()
        pygame.mixer.music.load("/home/pi/Documents/Codigo
Principal/Principal/audio/presion_normal.mp3")
        pygame.mixer.music.set_volume(0.45)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
    # generar lista de valores de signos vitales
    signosVit = [valTempStr, str(valBpm), str(valSpo2),
str(valPress)]
    guardaResp.saveResp(signosVit, 'signos')

    root.update()
```

Anexo 12 - Siglas

- ADC: Convertidor analógico a digital. Es un dispositivo que convierte una señal analógica variable en el tiempo a una representación de números digitales de la amplitud de esa señal.
- AMG: Acrónimo de Advanced MEMS Grid (Red de sistemas microelectrónicos avanzados).
- CMOS: Semiconductor complementario de oxido metálico. Es un elemento semiconductor utilizado en muchos ordenadores modernos y otros dispositivos electrónicos.
- GIMP: Es un programa de edición de imágenes digitales en forma de mapa de bits tanto dibujos como fotografías.
- CSV: Archivo de texto que tiene un formato específico que permite guardar los datos en un formato de tabla estructurada. Acrónimo de comma-separated values (valores separados por comas).
- DHT: Acrónimo que hace referencia a Digital relative Humidity and Temperature sensor (Sensor de temperatura y humedad relativa).
- HCSR-04: Módulo de sensor ultrasónico que determina la distancia de un objeto.
- I2C: Puerto y protocolo de comunicación serial, define la trama de datos y conexiones físicas para transferir bits entre 2 dispositivos digitales, acrónimo de Inter-Integrated Circuit (Circuito Inter Integrado).
- ESP: Acrónimo debido al fabricante chino Espressif Systems de Shanghái.
- GPIO: General Purpose Input Output. Sistema de entrada y salida de propósito general. Es decir, consta de una serie de pines o conexiones que se pueden usar como entradas o salidas para múltiples usos.
- FPGA: Integrado reconfigurable compuesto de interconexiones programables que combinan bloques lógicos programables, de memoria embebida y procesamiento

de señales digitales, acrónimo de Field Programmable Gate Arrays (Matriz de puertas lógicas programable).

- LIFA: Acrónimo de LabVIEW Interface For Arduino (Interfaz de LabVIEW para Arduino).
- MAX: Denominación otorgada por el fabricante para el circuito integrado de un sensor de ritmo cardíaco y oxigenación.
- MPX: Número de serie para sensores de silicio piezo-resistivos para medir presión de aire, dado por el fabricante “NXP”.
- PHPMYADMIN: Es una herramienta escrita en PHP (lenguaje de programación Hypertext Preprocessor) para manejar la administración de MySQL
- PuTTY: Pu se refiere a Port Unique, TTY se refiere a TELETYPE. Lo que sería Teletipo de puerto único.
- SARS-CoV-2: Acrónimo de Severe Acute Respiratory Syndrome Corona Virus 2 (Coronavirus de tipo 2 causante del síndrome respiratorio agudo severo).
- SPI: Es un protocolo de comunicación serial, acrónimo de Serial Peripheral Interface.
- SQL: Lenguaje de dominio especifico, diseñado para administrar y recuperar información, acrónimo de Structured Query Language (Lenguaje de consulta estructurada).
- SSH: Es un protocolo de administración remota, acrónimo de Secure Shell
- VNC: Es un software libre que permite ver las acciones del ordenador servidor remotamente a través de un ordenador cliente, acrónimo de Virtual Network Computing (Computación virtual en red).
- UART: Comunicación serial entre dos microcontroladores transmitiendo o recibiendo datos bit a bit, acrónimo de Universal Asynchronous receiver / transmitter (transmisor receptor asíncrono universal).
- UV: Rayos Ultravioleta.